Heart Disease Predictive Analysis

Sie Siong Wong

Contents

Prepared for:

City University of New York, School of Professional Studies

Prepared by:

Sie Siong Wong

Introduction

Cardiovascular diseases is the number 1 cause of death globally. The WHO records show that cardiovascular diseases accounted for 31% of all deaths worldwide in 2016. According to the CDC, nearly 6.2 million adults in the United States suffer heart failure, and in 2018 alone, heart failure was mentioned on 379,800 death certificates. Also, the treatment cost of health care services and medicines is costly and estimated about 31 billion in 2012. Early detection and management for heart disease could be effective in reducing the incidence of heart failure. The reason I have chosen this dataset to work with is because of how relevant this dataset is towards the real world. Heart disease is a condition which affects everyone and being able to detect potential heart failure ahead of time can save millions of lives.

For the homework 4, I'll be applying support vector machine and random forest algorithms with and without principal component analysis I have learned from this class to develop a good model to predict if a patient has heart disease and what factors can be attributed to heart disease most directly. The dataset for this homework **Heart Failure Prediction Dataset** can be found on Kaggle website. A total of 918 distinct individuals gathered from different countries and agencies in this dataset. Each observation in this dataset represents a person's health history, including their age, sex, cholesterol levels, etc.

Data Exploration

Attribute Information

Age: age of the patient [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 [mm Hg]
Cholesterol: serum cholesterol [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 class [1: heart disease, 0: Normal]

Read Data Source

```r
# Heart disease dataset
data <-
  read.csv("C:\\Users\\wongs34\\Documents\\School\\DATA 622\\Homework\\HW4\\heart.csv",
           header=TRUE,
           sep=",",
           check.names=FALSE)
```

Data Structure

Let's take a look to the first few rows of both data sets and its structure.

```r
print(head(data, 5))
```

```
##   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
##   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
```

```r
print(str(data))
```

```
## '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 ...
## NULL
```

This dataset contain 918 observations and 12 variables. There are 6 integer variables, 5 character variables, and 1 numeric variable.

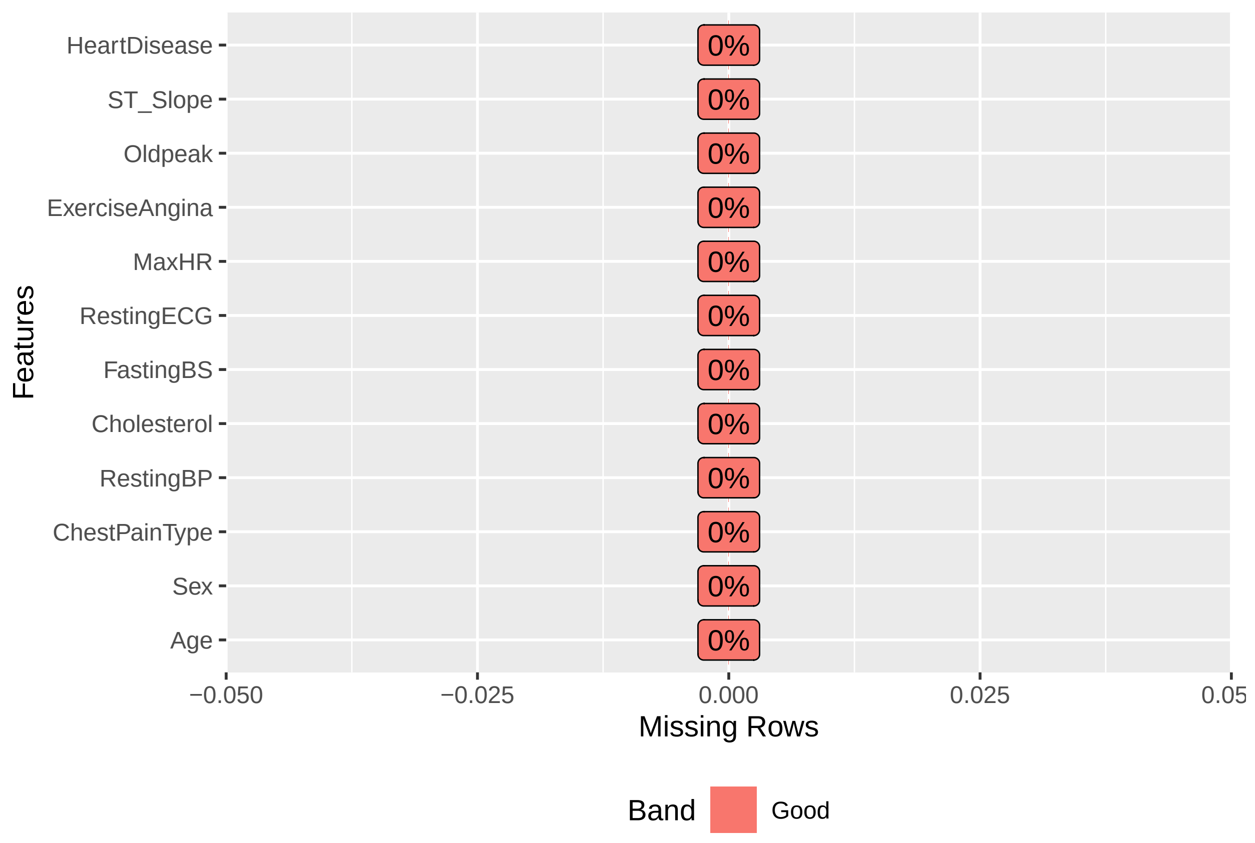
The response variable will be the **HeartDisease** and the rest of 11 variables are all predictors.

Summary Statistics

```r
# Summary statistic for numeric variables
data %>% dplyr::select(-c(Sex, ChestPainType, RestingECG, ExerciseAngina, ST_Slope)) %>%  psych::descri
```

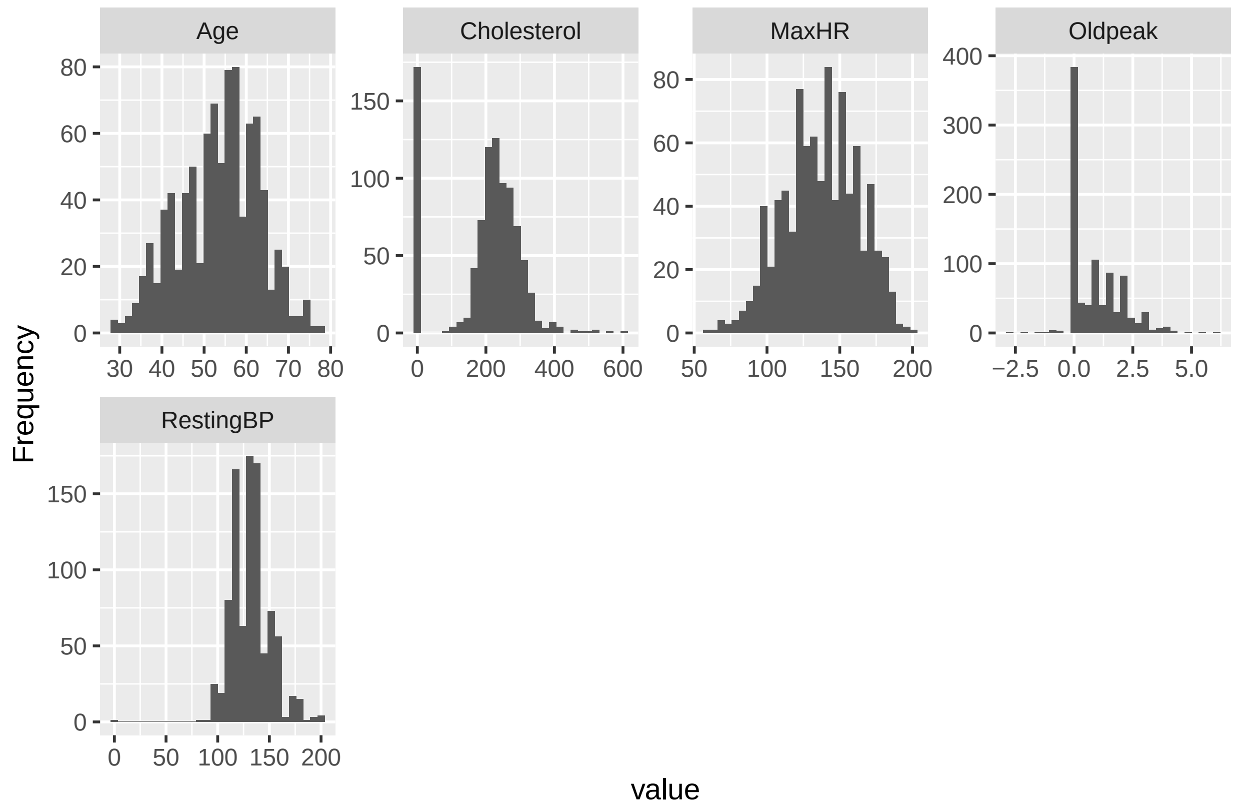
```
##              vars   n   mean     sd median trimmed   mad  min   max range  skew
## Age             1 918  53.51   9.43   54.0   53.71 10.38 28.0  77.0  49.0 -0.20
## RestingBP       2 918 132.40  18.51  130.0  131.50 14.83  0.0 200.0 200.0  0.18
## Cholesterol     3 918 198.80 109.38  223.0  204.41 68.20  0.0 603.0 603.0 -0.61
## FastingBS       4 918   0.23   0.42    0.0    0.17  0.00  0.0   1.0   1.0  1.26
## MaxHR           5 918 136.81  25.46  138.0  137.23 26.69 60.0 202.0 142.0 -0.14
## Oldpeak         6 918   0.89   1.07    0.6    0.74  0.89 -2.6   6.2   8.8  1.02
## HeartDisease    7 918   0.55   0.50    1.0    0.57  0.00  0.0   1.0   1.0 -0.21
##              kurtosis   se
## Age             -0.40 0.31
## RestingBP        3.23 0.61
## Cholesterol      0.10 3.61
## FastingBS       -0.41 0.01
## MaxHR           -0.46 0.84
## Oldpeak          1.18 0.04
## HeartDisease    -1.96 0.02
```

```r
# Check missing data for any variable
plot_missing(data)
```

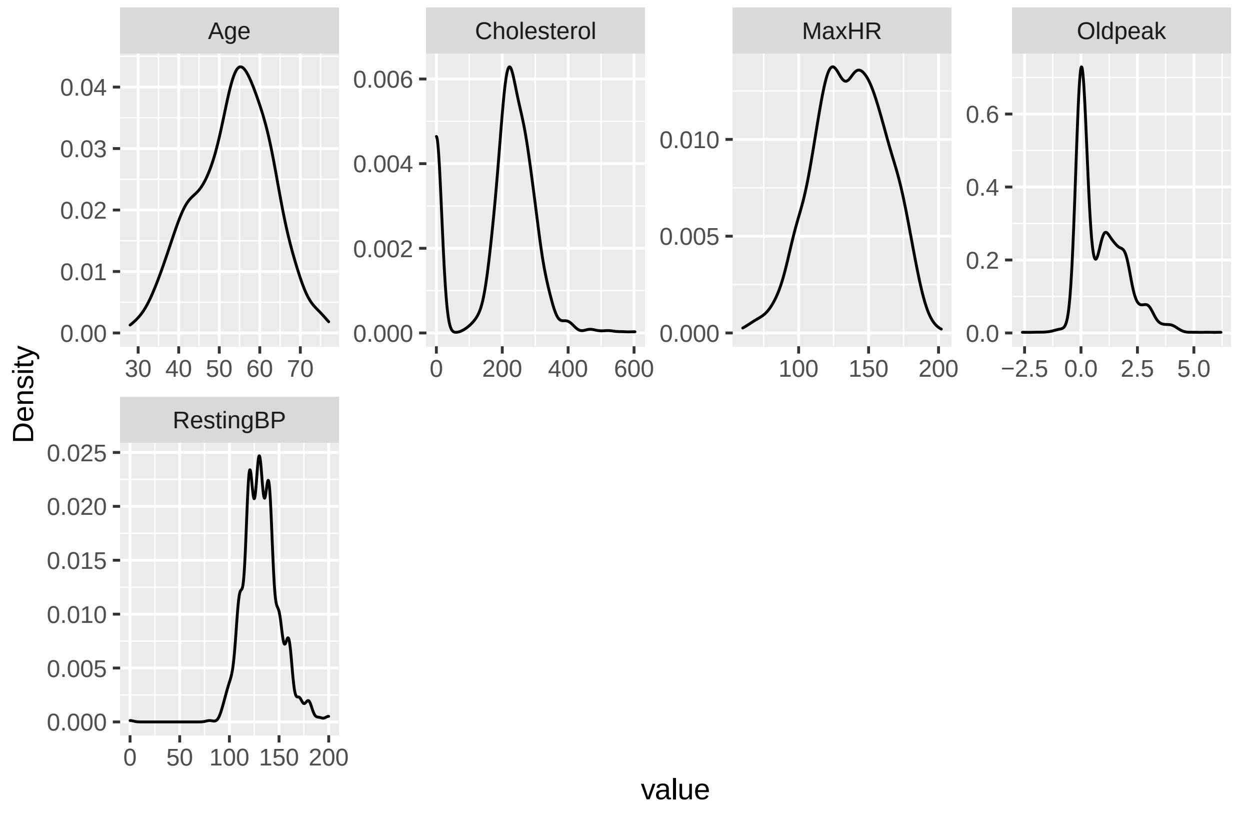


Quite a few of variables are high variance and skewed but 0% of missing data for this dataset. Let's continue to explore of this dataset using histogram and density plots.

```r
# Histogram and density plots
plot_histogram(data)
```



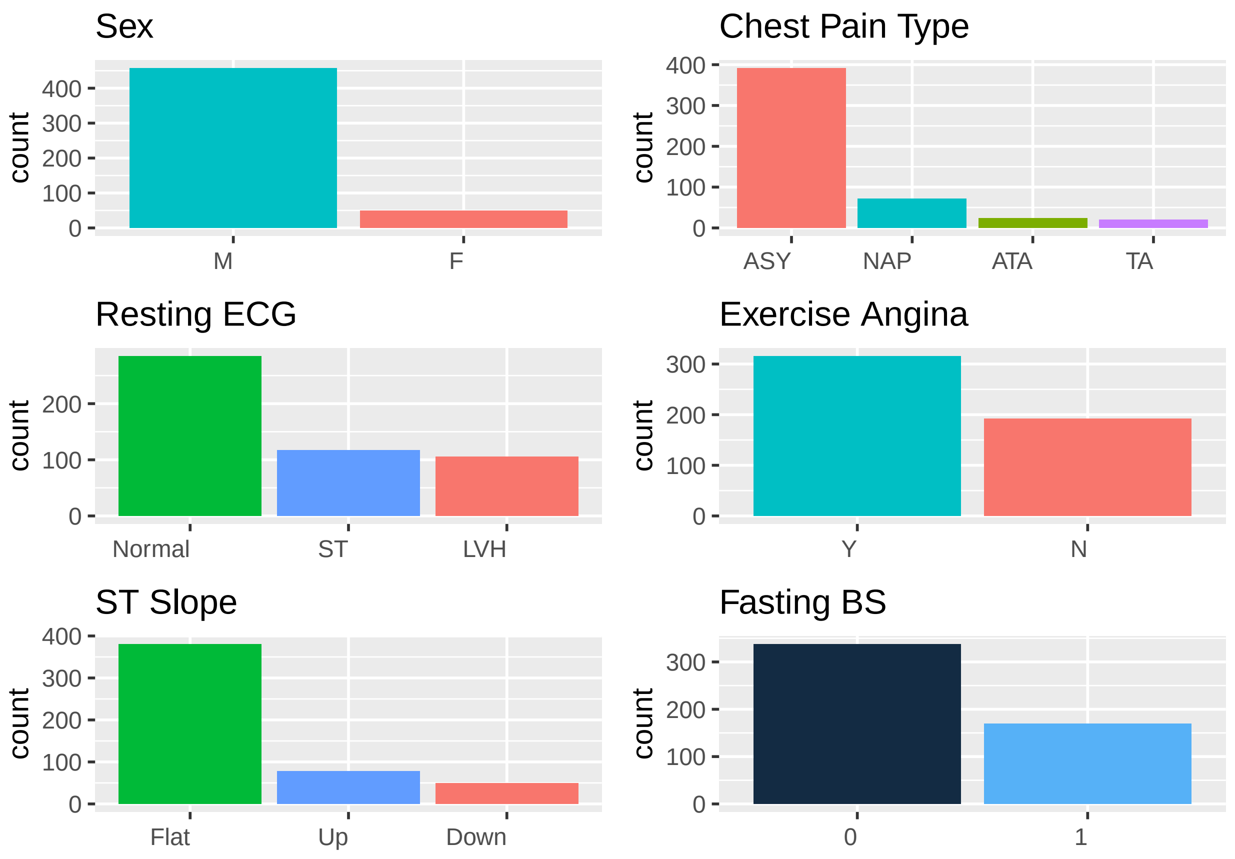
```r
plot_density(data)
```



From above histogram and density plots, we can notice that there are some zero values for Cholesterol and RestingBP variables. As Cholesterol and RestingBP measurement cannot be zero, in the data preparation section I'll consider to replace these zero values with mean or median or to impute them.

For variables which are character data type, I do count for these categorical type variables for which having heart disease. From below bar plots, we can notice that male patient is more likely to get heart disease than women patient. Also, most heart disease patients felt ASY type of chest pain, had exercise-induced angina, and flat slope of peak exercise, but non-diabetic (no fasting blood sugar) and normal ECG test.

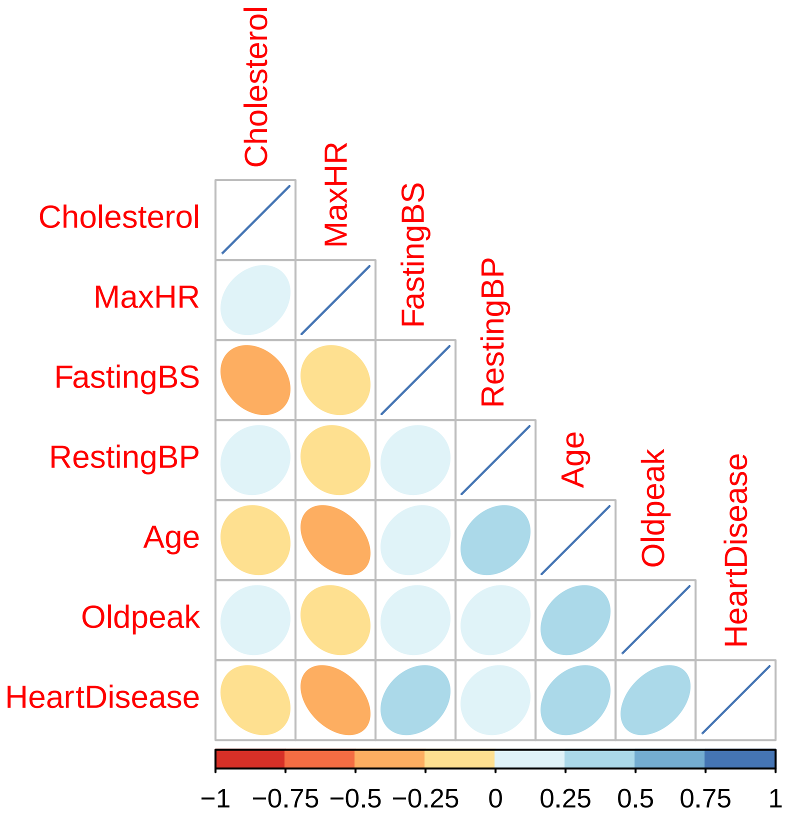
```r
# Create bar plot for each categorical type variables
sex <- data %>%
  dplyr::select(c(Sex, HeartDisease)) %>%
  filter(HeartDisease ==1) %>%
  group_by(HeartDisease, Sex) %>%
  summarise(count=n()) %>%
  ggplot(aes(x=reorder(Sex, -count), y=count, fill=Sex)) +
  geom_bar(stat="identity") +
  ggtitle("Sex") +
  theme(legend.position = "none",
        axis.title.x = element_blank(),
        axis.text.x = element_text(angle = 0, vjust = 0.5, hjust=1))

chestpain <- data %>%
  dplyr::select(c(ChestPainType, HeartDisease)) %>%
  filter(HeartDisease ==1) %>%
  group_by(HeartDisease, ChestPainType) %>%
  summarise(count=n()) %>%
  ggplot(aes(x=reorder(ChestPainType, -count), y=count, fill=ChestPainType)) +
  geom_bar(stat="identity") +
  ggtitle("Chest Pain Type") +
  theme(legend.position = "none",
        axis.title.x = element_blank(),
        axis.text.x = element_text(angle = 0, vjust = 0.5, hjust=1))

resting <- data %>%
  dplyr::select(c(RestingECG, HeartDisease)) %>%
  filter(HeartDisease ==1) %>%
  group_by(HeartDisease, RestingECG) %>%
  summarise(count=n()) %>%
  ggplot(aes(x=reorder(RestingECG, -count), y=count, fill=RestingECG)) +
  geom_bar(stat="identity") +
  ggtitle("Resting ECG") +
  theme(legend.position = "none",
        axis.title.x = element_blank(),
        axis.text.x = element_text(angle = 0, vjust = 0.5, hjust=1))

exercise <- data %>%
  dplyr::select(c(ExerciseAngina, HeartDisease)) %>%
  filter(HeartDisease ==1) %>%
  group_by(HeartDisease, ExerciseAngina) %>%
  summarise(count=n()) %>%
  ggplot(aes(x=reorder(ExerciseAngina, -count), y=count, fill= ExerciseAngina)) +
  geom_bar(stat="identity") +
  ggtitle("Exercise Angina") +
  theme(legend.position = "none",
        axis.title.x = element_blank(),
        axis.text.x = element_text(angle = 0, vjust = 0.5, hjust=1))

slope <- data %>%
  dplyr::select(c(ST_Slope, HeartDisease)) %>%
  filter(HeartDisease ==1) %>%
  group_by(HeartDisease, ST_Slope) %>%
  summarise(count=n()) %>%
  ggplot(aes(x=reorder(ST_Slope, -count), y=count, fill=ST_Slope)) +
  geom_bar(stat="identity") +
  ggtitle("ST Slope") +
  theme(legend.position = "none",
        axis.title.x = element_blank(),
        axis.text.x = element_text(angle = 0, vjust = 0.5, hjust=1))

diabetic <- data %>%
  dplyr::select(c(FastingBS, HeartDisease)) %>%
  filter(HeartDisease ==1) %>%
  group_by(HeartDisease, FastingBS) %>%
  summarise(count=n()) %>%
  ggplot(aes(x=reorder(FastingBS, -count), y=count, fill=FastingBS)) +
  geom_bar(stat="identity") +
  ggtitle("Fasting BS") +
  theme(legend.position = "none",
        axis.title.x = element_blank(),
        axis.text.x = element_text(angle = 0, vjust = 0.5, hjust=1))

ggarrange(sex, chestpain, resting, exercise, slope, diabetic,
          ncol = 2, nrow = 3)
```



Identifying Multicollinearity

The correlation in chart below shows some correlation among numeric predictors.

```
# Create correlation plot
correlation <- cor(dplyr::select(data,-c(Sex,
                                         ChestPainType,
                                         RestingECG,
                                         ExerciseAngina,
                                         ST_Slope)),
                   use = 'pairwise.complete.obs')

corrplot(correlation,
         method='ellipse',
         type = 'lower',
         order = 'hclust',
         col=brewer.pal(n=8, name="RdYlBu"))
```



We can do a correlation test for each continuous predictor by using the Pearson method and at 95% confidence level to determine if any correlation among predictors is significant. From the table below, we can see most pairwise combinations' p-value are less than 5% significant level. This indicates those pairwise combinations are significantly correlated. Note that the effect of collinearity on prediction is less serious. The accuracy of the prediction depends on the distance from the observed data. Collinear data only covers very small fraction of the the predictor space. However, principal component analysis can help to solve this issue in later section.

```
chol <- data %>%
  cor_test(vars="Cholesterol",
           vars2=c("MaxHR","FastingBS","RestingBP","Age","Oldpeak"),
           method="pearson")
maxhr <- data %>%
  cor_test(vars="MaxHR",
           vars2=c("Cholesterol","FastingBS","RestingBP","Age","Oldpeak"),
           method="pearson")
fastbs <- data %>%
  cor_test(vars="FastingBS",
           vars2=c("Cholesterol","MaxHR","RestingBP","Age","Oldpeak"),
           method="pearson")
restbp <- data %>%
  cor_test(vars="RestingBP",
           vars2=c("Cholesterol","MaxHR","FastingBS","Age","Oldpeak"),
           method="pearson")
age <- data %>%
  cor_test(vars="Age",
           vars2=c("Cholesterol","MaxHR","FastingBS","RestingBP","Oldpeak"),
           method="pearson")
oldp <- data %>%
  cor_test(vars="Oldpeak",
           vars2=c("Cholesterol","MaxHR","FastingBS","RestingBP","Age"),
           method="pearson")
combine_df <- bind_rows(chol, maxhr, fastbs, restbp, age, oldp) %>%
  select(c(var1,var2, p))
combine_df[order(-combine_df$p),]
```

```
## # A tibble: 30 x 3
##    var1        var2              p
##    <chr>       <chr>         <dbl>
##  1 Cholesterol Oldpeak     0.129
##  2 Oldpeak     Cholesterol 0.129
##  3 FastingBS   Oldpeak     0.111
##  4 Oldpeak     FastingBS   0.111
##  5 FastingBS   RestingBP   0.0335
##  6 RestingBP   FastingBS   0.0335
##  7 Cholesterol Age         0.00386
##  8 Age         Cholesterol 0.00386
##  9 Cholesterol RestingBP   0.00221
## 10 RestingBP   Cholesterol 0.00221
## # ... with 20 more rows
```

Data Preparation

Data Type Transformation

First of all, we need to transform data type for some of the variables to be used in model building.

```
# Convert character type to factor
data <- data %>%
  mutate_at(c('Sex','ChestPainType','RestingECG', 'FastingBS', 'ExerciseAngina', 'ST_Slope', 'HeartDise

# Convert integer type variables to numerical
data <- data %>%
  mutate_at(c('RestingBP','Age', 'Cholesterol','MaxHR'),
            funs(as.numeric(.)))
```

Outliers

As we've seen in the data exploration section, these two variables **Cholesterol** and **Resting BP** which measurement cannot be zero. I'll be using the median to fix all the zero values as the median is less susceptible to outliers.

```
# Replace zero with median value
data <- data %>%
  mutate(Cholesterol = ifelse(Cholesterol==0,
                              median(Cholesterol, na.rm = TRUE),
                              Cholesterol)) %>%
  mutate(Cholesterol = ifelse(RestingBP==0,
                              median(Cholesterol, na.rm = TRUE),
                              RestingBP))
```

Split Dataset

I split the dataset by allocating 70% of the dataset to the training set and 30% to the test set. The training set will be used to build models and the test set will be used to evaluate the performance of the models.

```
set.seed(123)

# Split the data into train and test sets
train_index <- createDataPartition(data$HeartDisease, p = .7, list = FALSE, times = 1)
train_data <- data[train_index,]
test_data <- data[-train_index,]
```

Balancing Training Set

Because it's classification analysis, it's important to check the class distributions between all data sets to make sure that they are similar.

```
round(prop.table(table(dplyr::select(data, HeartDisease), exclude = NULL)), 2)*100
```

```
##
##  0  1
## 45 55
```

```
round(prop.table(table(dplyr::select(train_data, HeartDisease), exclude = NULL)), 2)*100
```

```
##
##  0  1
## 45 55
```

```
round(prop.table(table(dplyr::select(test_data, HeartDisease), exclude = NULL)), 2)*100
```

```
##
##  0  1
## 45 55
```

Class imbalance is a common problem when working with real-world data. It degrades the performance of a machine learning model because it biases the model toward the majority class at the expense of the minority class. Above we see a bit of imbalance in the training data sets. We need to balance the training set prior to the modeling process. Note that this applies only to the training set. To do that, we use a function called SMOTE() in the **DMwR** package provides us to balance our training data.

```r
set.seed(1234)

# Balance training data sets
train_data <- SMOTE(HeartDisease ~ ., data=train_data,
                    perc.over=100,
                    perc.under=200,
                    k=5,
                    learner=NULL)
```

Now, we have the training data set classes reached a 50/50 balance.

```r
round(prop.table(table(dplyr::select(train_data, HeartDisease), exclude = NULL)), 4)*100
```

```
##
##  0  1
## 50 50
```

Build Model

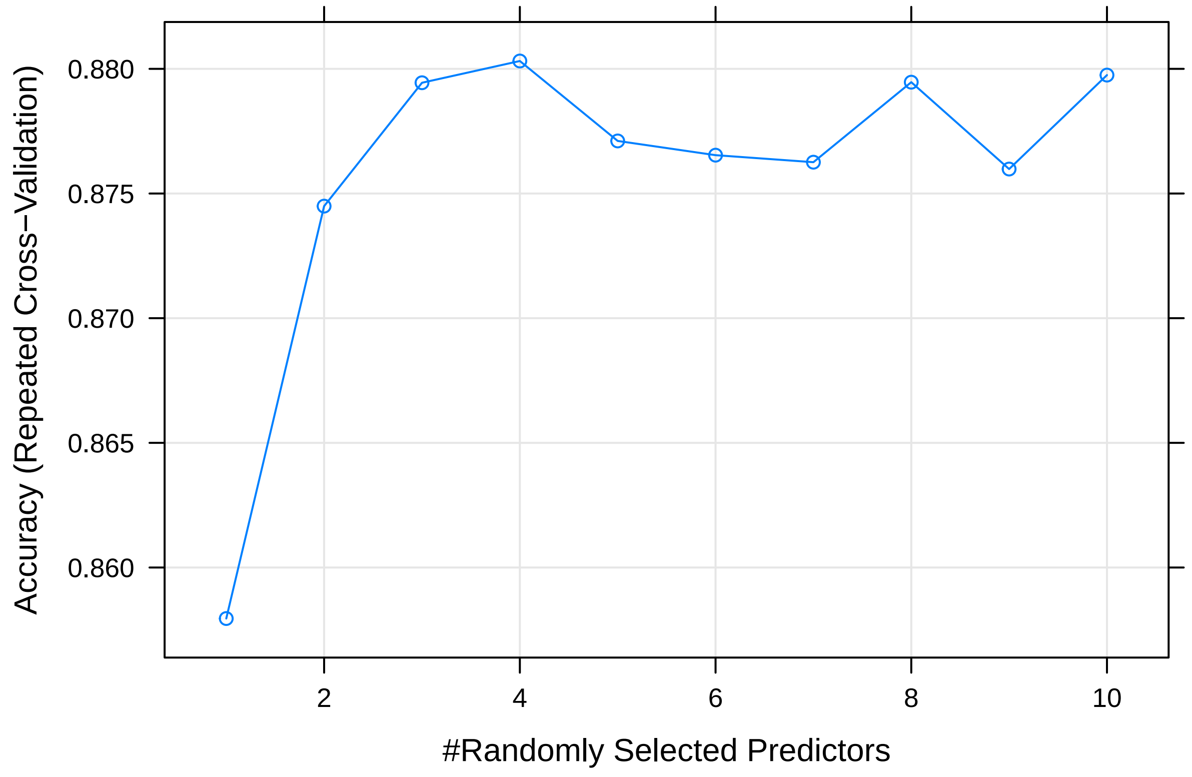
Random Forest

I build a random forest model by using the train() function from the caret package and random forest algorithm from the RandomForest package.

```
set.seed(977)

# Build and train random forest model
control <- trainControl(method="repeatedcv", number=10, repeats=3, search="grid")
tunegrid <- expand.grid(.mtry=c(1:10))
rf_mod <- train(x=select(train_data,-HeartDisease),
                y=train_data$HeartDisease,
                method="rf",
                metric="Accuracy",
                tuneGrid=tunegrid,
                trControl=control,
                ntree=120,
                nodesize=10,
                importance = TRUE)

# Print and plot the result
print(rf_mod)
```

```
## Random Forest
##
## 1148 samples
##   11 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 1033, 1033, 1033, 1034, 1034, 1032, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    1    0.8579541  0.7158705
##    2    0.8744940  0.7489370
##    3    0.8794418  0.7588359
##    4    0.8803140  0.7605829
##    5    0.8771077  0.7541663
##    6    0.8765380  0.7530400
##    7    0.8762558  0.7524760
##    8    0.8794643  0.7589010
##    9    0.8759810  0.7519279
##   10    0.8797493  0.7594561
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 4.
```

```
plot(rf_mod)
```



Using the grid search of algorithm parameters to try, we can see above result shows that the best performance happens at mtry = 4 with accuracy of 88.03%.

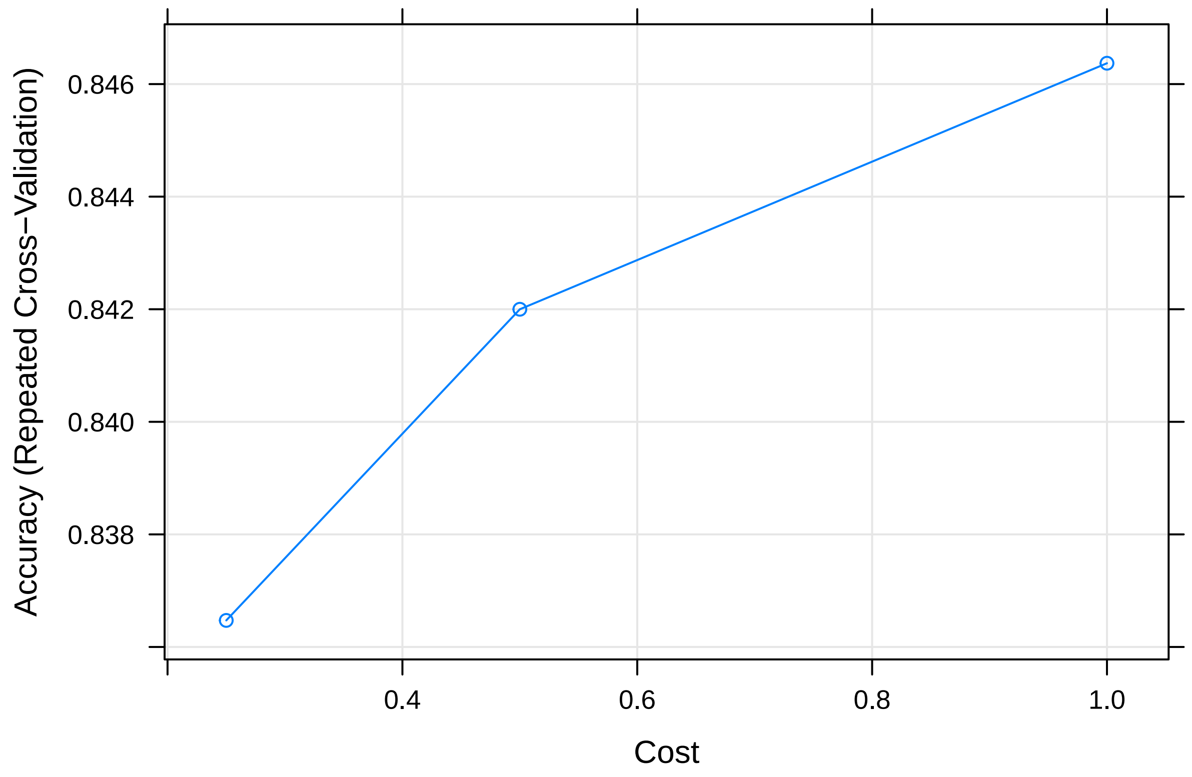
Support Vector Machine

```
set.seed(343)

# Build and train support vector machine model
control <- trainControl(method="repeatedcv", number=10, repeats=3, search="grid")
tunegrid <- expand.grid(.mtry=c(1:10))
svm_mod <- train(HeartDisease ~.,
                 data = train_data,
                 method = "svmRadial",
                 trControl = control,
                 tunegrid = tunegrid)

# Print and plot the result
print(svm_mod)
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
```

```
## 1148 samples
##   11 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 1033, 1032, 1033, 1033, 1033, 1033, ...
## Resampling results across tuning parameters:
##
##   C     Accuracy   Kappa
##   0.25  0.8364718  0.6729501
##   0.50  0.8419995  0.6840044
##   1.00  0.8463702  0.6927351
##
## Tuning parameter 'sigma' was held constant at a value of 0.0457384
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.0457384 and C = 1.
```

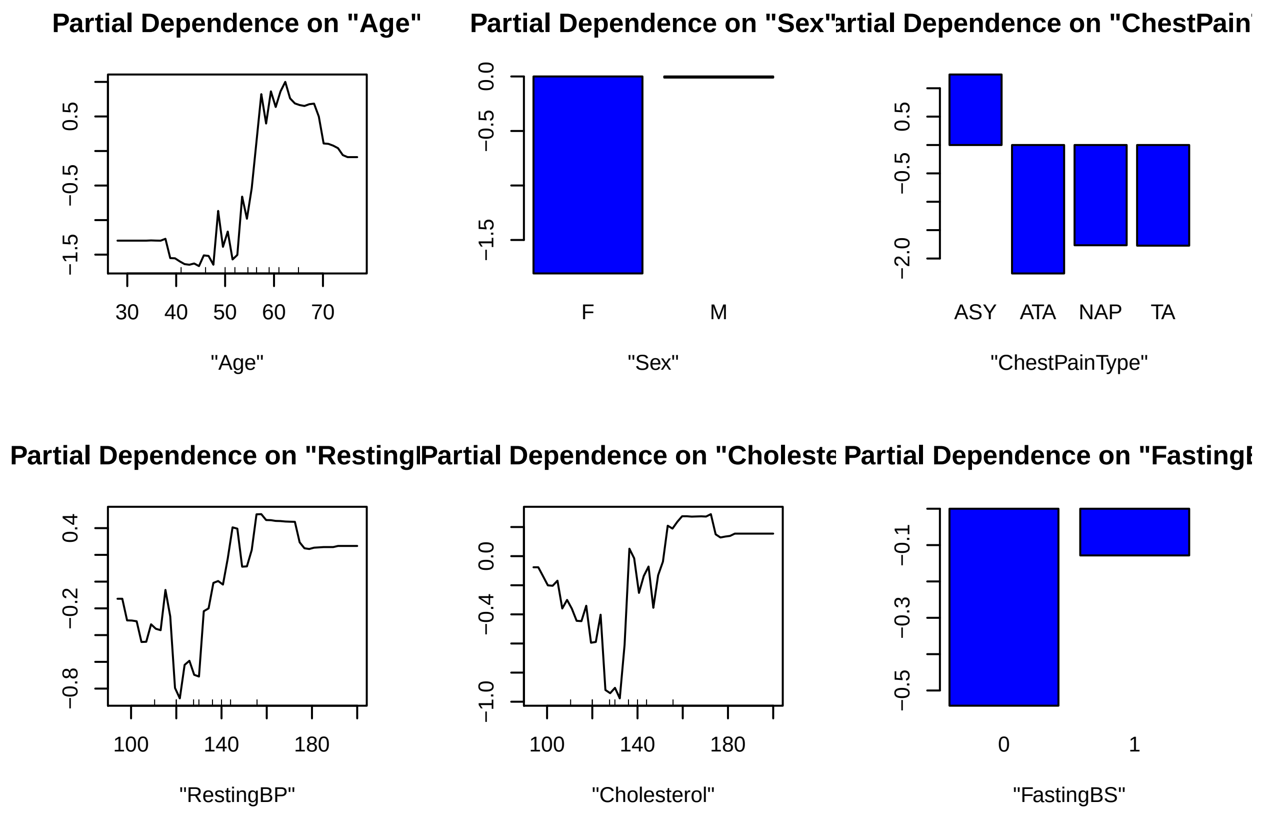
```r
plot(svm_mod)
```



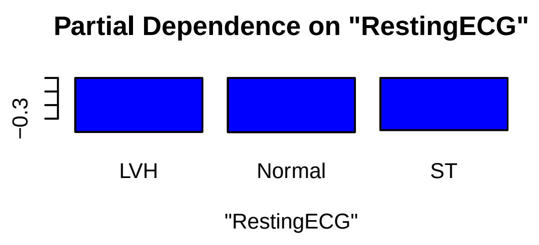
The best tune parameter values for Cost is 1. This is also the default Cost value given in the configuration. The best sigma value is 0.0457384 and was held constant through out the cross validation.

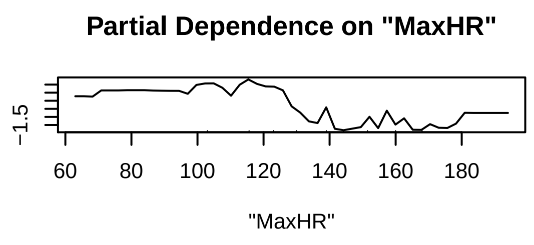
I use the partial dependence plot to visualize how likely the values of each variable contributed to the chance of survival. Negative value is less likely, positive value is more likely, and zero implies no average impact on class probability according to the model. Because the partial dependence plots will look similar for both SVM and random forest models, I choose to use random forest final model.

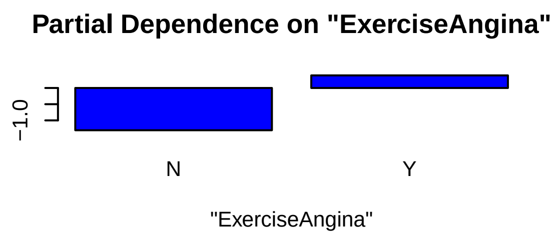
```r
par(mfrow=c(2,3))
partialPlot(rf_mod$finalModel,train_data,'Age', '1')
partialPlot(rf_mod$finalModel,train_data,'Sex', '1')
partialPlot(rf_mod$finalModel,train_data,'ChestPainType', '1')
partialPlot(rf_mod$finalModel,train_data,'RestingBP', '1')
partialPlot(rf_mod$finalModel,train_data,'Cholesterol', '1')
partialPlot(rf_mod$finalModel,train_data,'FastingBS', '1')
```

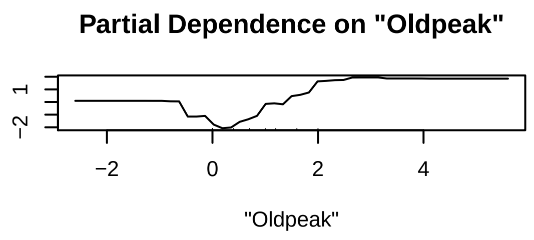


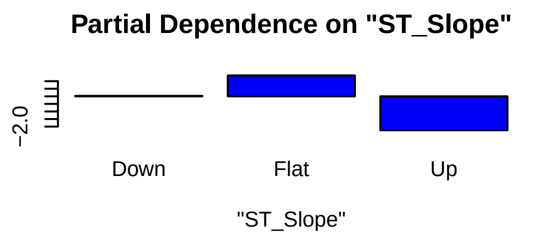
```r
par(mfrow=c(3,2))
partialPlot(rf_mod$finalModel,train_data,'RestingECG', '1')
partialPlot(rf_mod$finalModel,train_data,'MaxHR', '1')
partialPlot(rf_mod$finalModel,train_data,'ExerciseAngina', '1')
partialPlot(rf_mod$finalModel,train_data,'Oldpeak', '1')
partialPlot(rf_mod$finalModel,train_data,'ST_Slope', '1')
```











From the partial plots above, we can see that older male patients with high blood pressure, cholesterol, chest pain type of "ASY", resting ECG type of "ST", ST Slope type of "Flat", and has exercise angina,low MaxHR and high Oldpeak, will have higher chance of getting heart disease.

Principal Component Analysis

PCA (Principal Component Analysis) combines the highly correlated variables into a set of uncorrelated variables to effectively eliminate multicollinearity between features.

Since PCA only works for numerical, I need to convert all factor variables to numeric first before I can apply PCA for dimensionality reduction.

```r
# Extract numeric variables
numeric_set_tr <- train_data %>% keep(is.numeric)
factor_set_tr <- train_data %>% keep(is.factor)
numeric_set_ts <- test_data %>% keep(is.numeric)
factor_set_ts <- test_data %>% keep(is.factor)

# Convert factor set to numerical
factor_set_tr$Sex <- as.numeric(factor_set_tr$Sex)
factor_set_tr$ChestPainType <- as.numeric(factor_set_tr$ChestPainType)
factor_set_tr$FastingBS <- as.numeric(factor_set_tr$FastingBS)
factor_set_tr$RestingECG <- as.numeric(factor_set_tr$RestingECG)
factor_set_tr$ExerciseAngina <- as.numeric(factor_set_tr$ExerciseAngina)
factor_set_tr$ST_Slope <- as.numeric(factor_set_tr$ST_Slope)
```

```
factor_set_ts$Sex <- as.numeric(factor_set_ts$Sex)
factor_set_ts$ChestPainType <- as.numeric(factor_set_ts$ChestPainType)
factor_set_ts$FastingBS <- as.numeric(factor_set_ts$FastingBS)
factor_set_ts$RestingECG <- as.numeric(factor_set_ts$RestingECG)
factor_set_ts$ExerciseAngina <- as.numeric(factor_set_ts$ExerciseAngina)
factor_set_ts$ST_Slope <- as.numeric(factor_set_ts$ST_Slope)

# Combine both sets
train_data_pca <- cbind(numeric_set_tr, factor_set_tr)
test_data_pca <- cbind(numeric_set_ts, factor_set_ts)
```

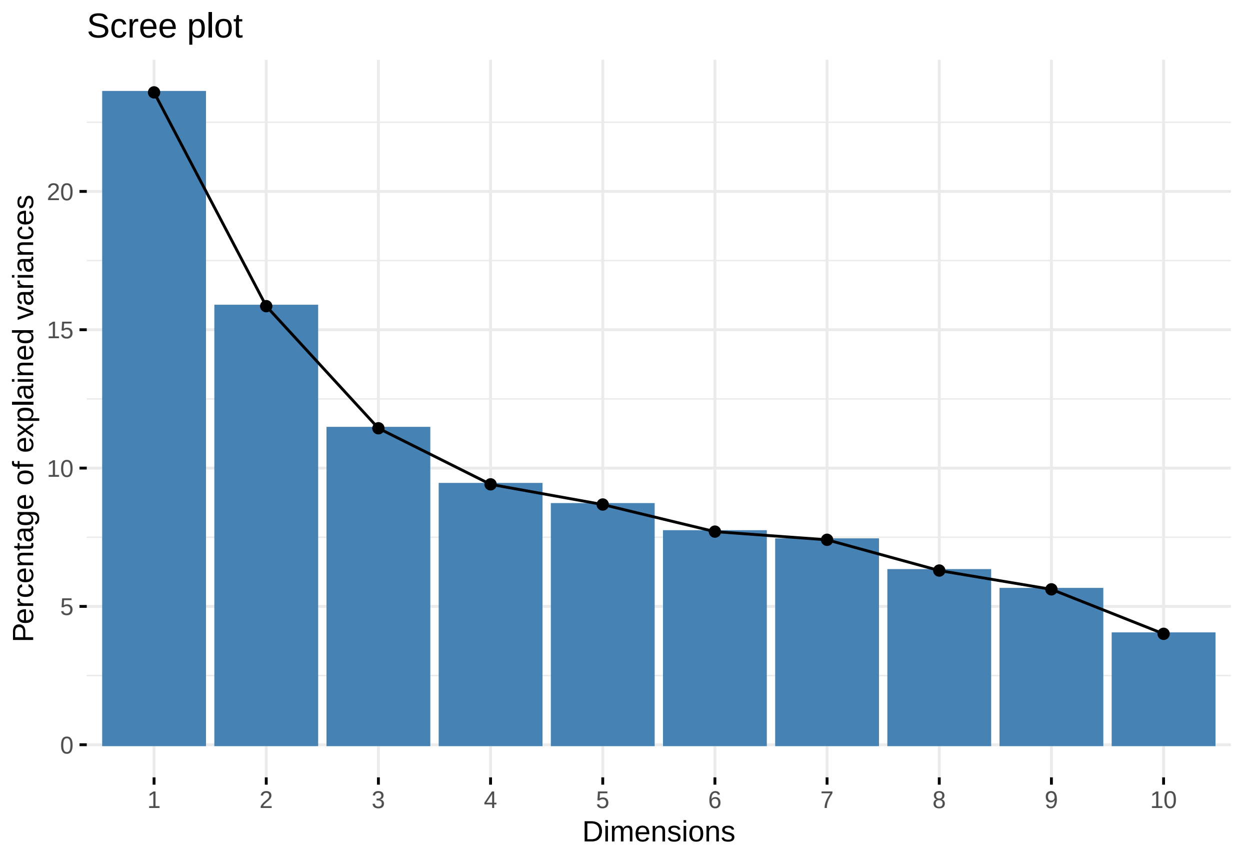
Now, we can do PCA using the prcomp() function from stats package.

```
pca_mod <- prcomp(select(train_data_pca, -HeartDisease),
                 center = TRUE,
                 scale. = TRUE)

summary(pca_mod)
```

```
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5     PC6     PC7
## Standard deviation     1.6105 1.3205 1.1217 1.01753 0.97728 0.92053 0.90268
## Proportion of Variance 0.2358 0.1585 0.1144 0.09412 0.08683 0.07703 0.07408
## Cumulative Proportion  0.2358 0.3943 0.5087 0.60283 0.68966 0.76669 0.84077
##                            PC8     PC9    PC10      PC11
## Standard deviation     0.83225 0.78605 0.66410 5.691e-16
## Proportion of Variance 0.06297 0.05617 0.04009 0.000e+00
## Cumulative Proportion  0.90374 0.95991 1.00000 1.000e+00
```

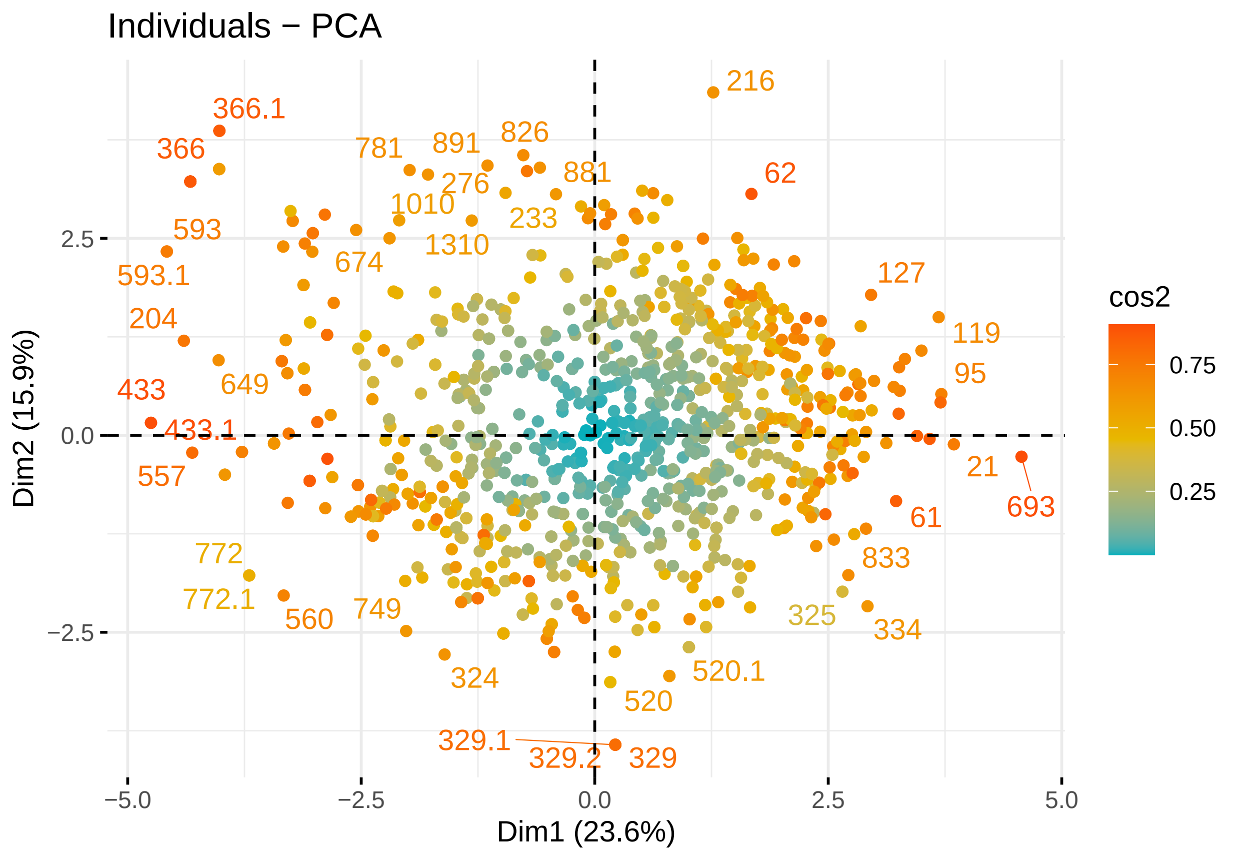
From the summary above, we can see that the first 3 principal components capture the most variability. The rest of the variability spread quite equally among the rest of the principal components. We can also visualize this using the **factoextra** package. Below scree plot shows the percentage of variances explained by each principal component.

```
fviz_eig(pca_mod)
```



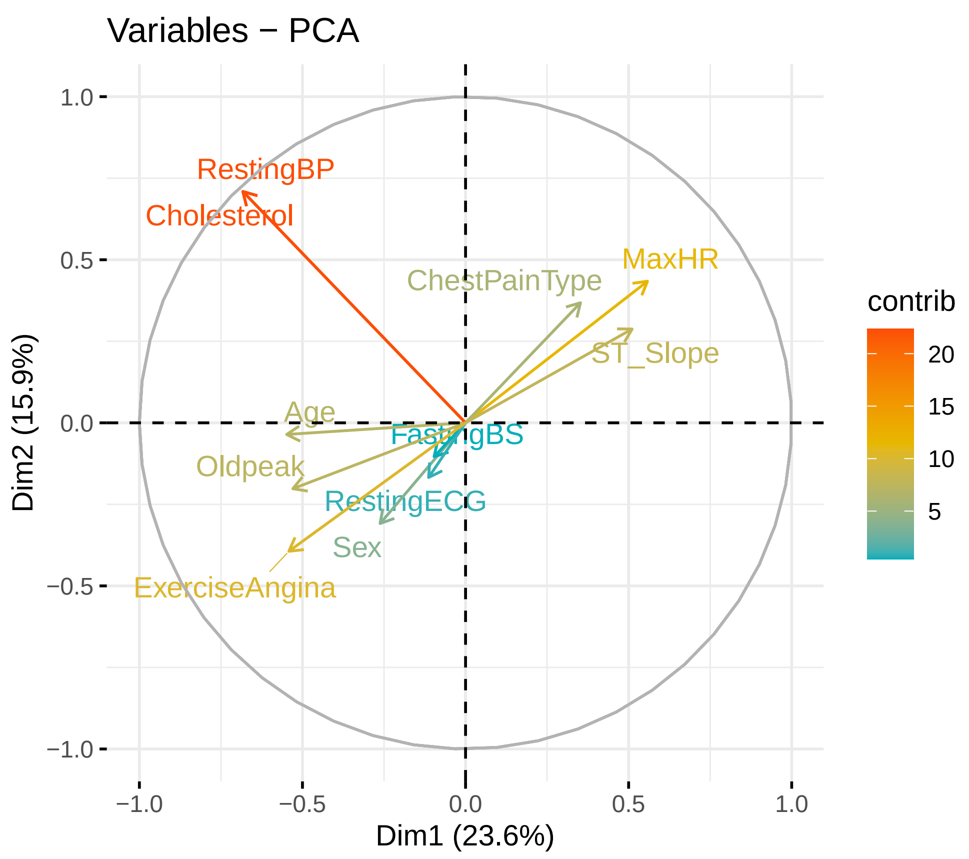
We can also do graph of individuals using the same package as above to visualize individuals with similar profile are grouped together.

```
fviz_pca_ind(pca_mod,
             col.ind = "cos2",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE
             )
```



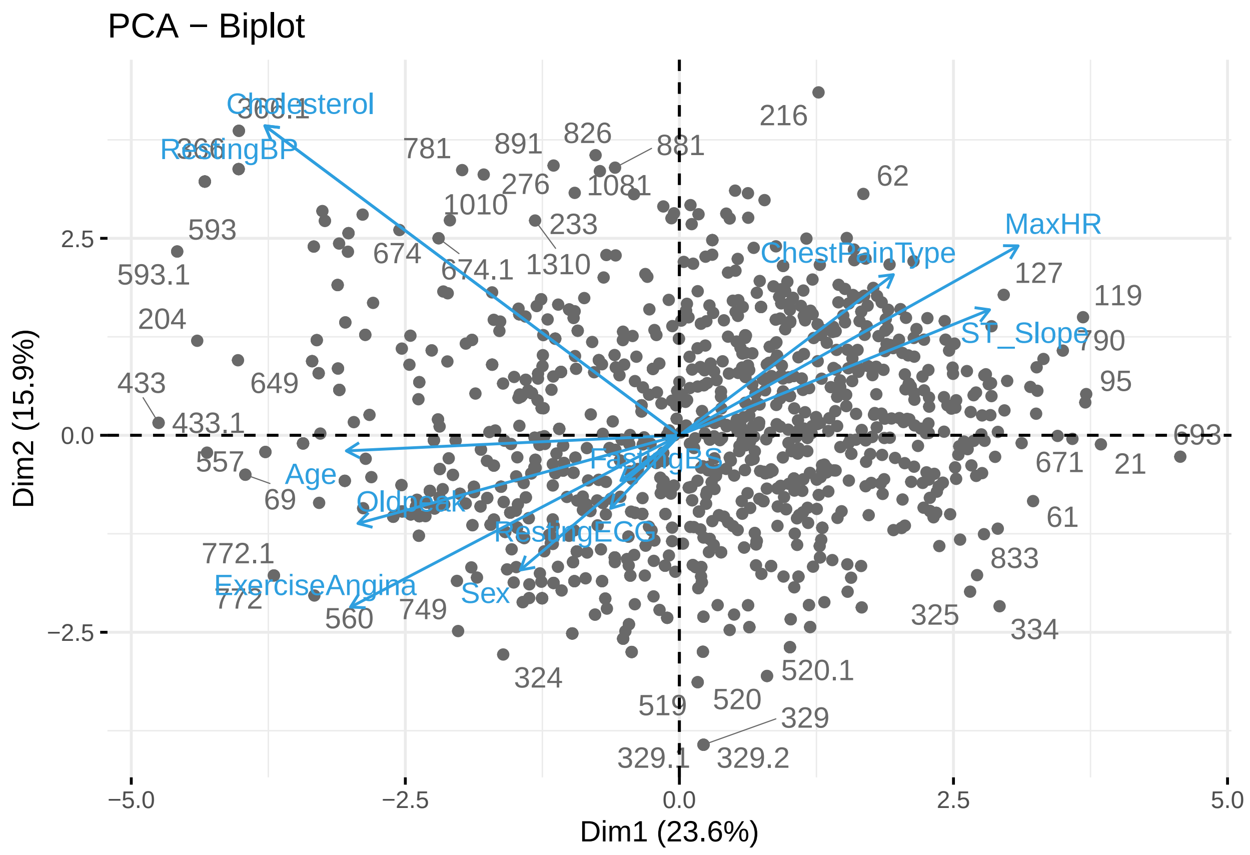
In addition to graph of individuals, there is also graph of variables where positive correlated variables point to the same side of the plot. Negative correlated variables point to opposite sides of the graph.

```
fviz_pca_var(pca_mod,
             col.var = "contrib",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE
             )
```



Last but not least, is the biplot of individuals and variables.

```
fviz_pca_biplot(pca_mod, repel = TRUE,
                col.var = "#2E9FDF",
                col.ind = "#696969"
                )
```



Then, we need to prepare train and test principal components data sets for rebuilding random forest model.

```r
# Prediction with principal components
pca_pred_tr <- predict(pca_mod, train_data_pca)
pca_pred_ts <- predict(pca_mod, test_data_pca)

# Add the heart disease column information into pca_pred_tr
pca_pred_tr <- data.frame(pca_pred_tr, train_data_pca[,"HeartDisease"]) %>%
  rename(HeartDisease = train_data_pca....HeartDisease..)
pca_pred_ts <- data.frame(pca_pred_ts, test_data_pca[,"HeartDisease"]) %>%
  rename(HeartDisease = test_data_pca....HeartDisease..)
```

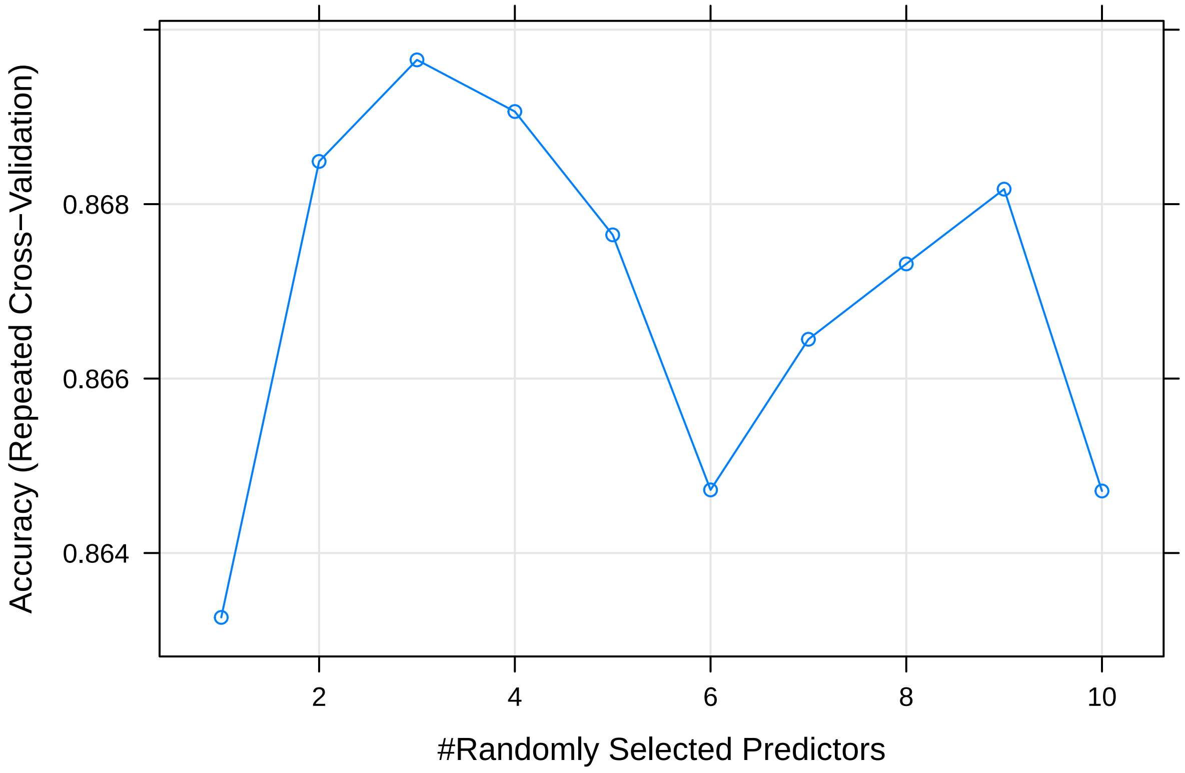
Rebuild Models Using PCs

Random Forest After try with different number principal components, combination of PC1, PC2, PC3, PC4, PC5, PC6, and PC7 will give the highest accuracy.

```r
set.seed(444)

# Rebuild random forest model using principal components data set
control <- trainControl(method="repeatedcv", number=10, repeats=3, search="grid")
tunegrid <- expand.grid(.mtry=c(1:10))
rf_mod_pca <- train(HeartDisease~PC1+PC2+PC3+PC4+PC5+PC6+PC7,
                    data=pca_pred_tr,
                    method="rf",
                    metric="Accuracy",
                    tuneGrid=tunegrid,
                    trControl=control,
                    ntree = 120,
                    nodesize=10,
                    importance = TRUE)

# Print and plot the result
print(rf_mod_pca)
```

```
## Random Forest
##
## 1148 samples
##    7 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 1034, 1032, 1033, 1033, 1033, 1034, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    1    0.8632616  0.7265417
##    2    0.8684892  0.7369997
##    3    0.8696538  0.7393241
##    4    0.8690616  0.7381482
##    5    0.8676477  0.7352974
##    6    0.8647237  0.7294523
##    7    0.8664503  0.7329059
##    8    0.8673149  0.7346380
##    9    0.8681720  0.7363541
##   10    0.8647111  0.7294384
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 3.
```

```r
plot(rf_mod_pca)
```



The best performance happens at mtry = 3 with accuracy of 86.97%.

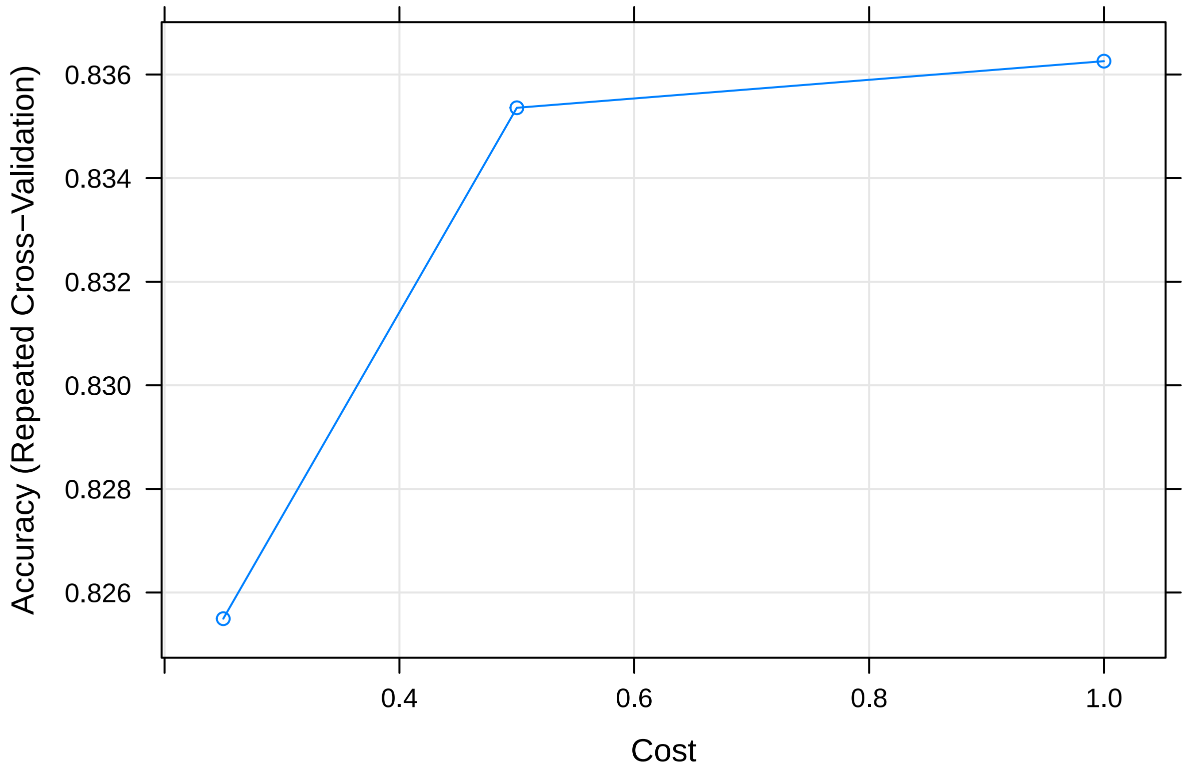
Support Vector Machine After try with different number principal components, combination of PC1, PC2, PC3, PC4, PC5, PC6, PC7, PC8, PC9 will give the highest accuracy.

```
set.seed(919)

# Rebuild random forest model using principal components data set
control <- trainControl(method="repeatedcv", number=10, repeats=3, search="grid")
tunegrid <- expand.grid(.mtry=c(1:10))
svm_mod_pca <- train(HeartDisease~PC1+PC2+PC3+PC4+PC5+PC6+PC7+PC8+PC9,
                     data = pca_pred_tr,
                     method = "svmRadial",
                     trControl = control,
                     tunegrid = tunegrid)

# Print and plot the result
print(svm_mod_pca)
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 1148 samples
##    9 predictor
##    2 classes: '0', '1'
##
```

```
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 1033, 1033, 1032, 1034, 1034, 1033, ...
## Resampling results across tuning parameters:
##
##   C     Accuracy   Kappa
##   0.25  0.8254945  0.6509699
##   0.50  0.8353575  0.6706915
##   1.00  0.8362575  0.6725008
##
## Tuning parameter 'sigma' was held constant at a value of 0.07917633
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.07917633 and C = 1.
```

```r
plot(svm_mod_pca)
```



The best sigma value is 0.07917633 by using principal components and was held constant through out the cross validation.

Model Evaluation

Now that we have built two random forest models. One model built using original data and another model built using principal components. Now, let's generate predictions against the test data to evaluate both models' performance. To do that, I create a confusion matrix and use its values to derive our model's predictive accuracy.

Random Forest Model without PCs

```r
# Evaluate random forest model performance using normal test data set
pred_rf_mod <- predict(rf_mod, test_data)
cm_rf <- caret::confusionMatrix(pred_rf_mod, test_data$HeartDisease, positive="1")
cm_rf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 111  28
##          1  12 124
##
##                Accuracy : 0.8545
##                  95% CI : (0.8072, 0.894)
##     No Information Rate : 0.5527
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.7094
##
##  Mcnemar's Test P-Value : 0.01771
##
##             Sensitivity : 0.8158
##             Specificity : 0.9024
##          Pos Pred Value : 0.9118
##          Neg Pred Value : 0.7986
##              Prevalence : 0.5527
##          Detection Rate : 0.4509
##    Detection Prevalence : 0.4945
##       Balanced Accuracy : 0.8591
##
##        'Positive' Class : 1
##
```

Support Vector Machine Model without PCs

```r
# Evaluate random forest model performance using normal test data set
pred_svm_mod <- predict(svm_mod, test_data)
cm_svm <- caret::confusionMatrix(pred_svm_mod, test_data$HeartDisease, positive="1")
cm_svm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 114  28
##          1   9 124
##
##                Accuracy : 0.8655
##                  95% CI : (0.8193, 0.9035)
##     No Information Rate : 0.5527
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7318
##
##  Mcnemar's Test P-Value : 0.003085
##
##             Sensitivity : 0.8158
##             Specificity : 0.9268
##          Pos Pred Value : 0.9323
##          Neg Pred Value : 0.8028
##              Prevalence : 0.5527
##          Detection Rate : 0.4509
##    Detection Prevalence : 0.4836
##       Balanced Accuracy : 0.8713
##
##        'Positive' Class : 1
##
```

Random Forest Model with PCs

```
# Evaluate random forest model performance using pca test data set
pred_rf_mod_pca <- predict(rf_mod_pca, pca_pred_ts)
cm_rf_pca <- caret::confusionMatrix(pred_rf_mod_pca, test_data_pca$HeartDisease, positive="1")
cm_rf_pca
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 120  44
##          1   3 108
##
##                Accuracy : 0.8291
##                  95% CI : (0.7793, 0.8716)
##     No Information Rate : 0.5527
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.665
##
##  Mcnemar's Test P-Value : 5.392e-09
##
##             Sensitivity : 0.7105
##             Specificity : 0.9756
##          Pos Pred Value : 0.9730
##          Neg Pred Value : 0.7317
##              Prevalence : 0.5527
##          Detection Rate : 0.3927
##    Detection Prevalence : 0.4036
##       Balanced Accuracy : 0.8431
##
##        'Positive' Class : 1
##
```

Support Vector Machine Model with PCs

```
# Evaluate random forest model performance using pca test data set
pred_svm_mod_pca <- predict(svm_mod_pca, pca_pred_ts)
cm_svm_pca <- caret::confusionMatrix(pred_svm_mod_pca, test_data_pca$HeartDisease, positive="1")
cm_svm_pca
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 110  25
##          1  13 127
##
##                Accuracy : 0.8618
##                  95% CI : (0.8153, 0.9003)
##     No Information Rate : 0.5527
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.7231
##
##  Mcnemar's Test P-Value : 0.07435
##
##             Sensitivity : 0.8355
##             Specificity : 0.8943
##          Pos Pred Value : 0.9071
##          Neg Pred Value : 0.8148
##              Prevalence : 0.5527
##          Detection Rate : 0.4618
##    Detection Prevalence : 0.5091
##       Balanced Accuracy : 0.8649
##
##        'Positive' Class : 1
##
```

Summary

Accuracy

Below is the summary table for all models accuracy measurement.

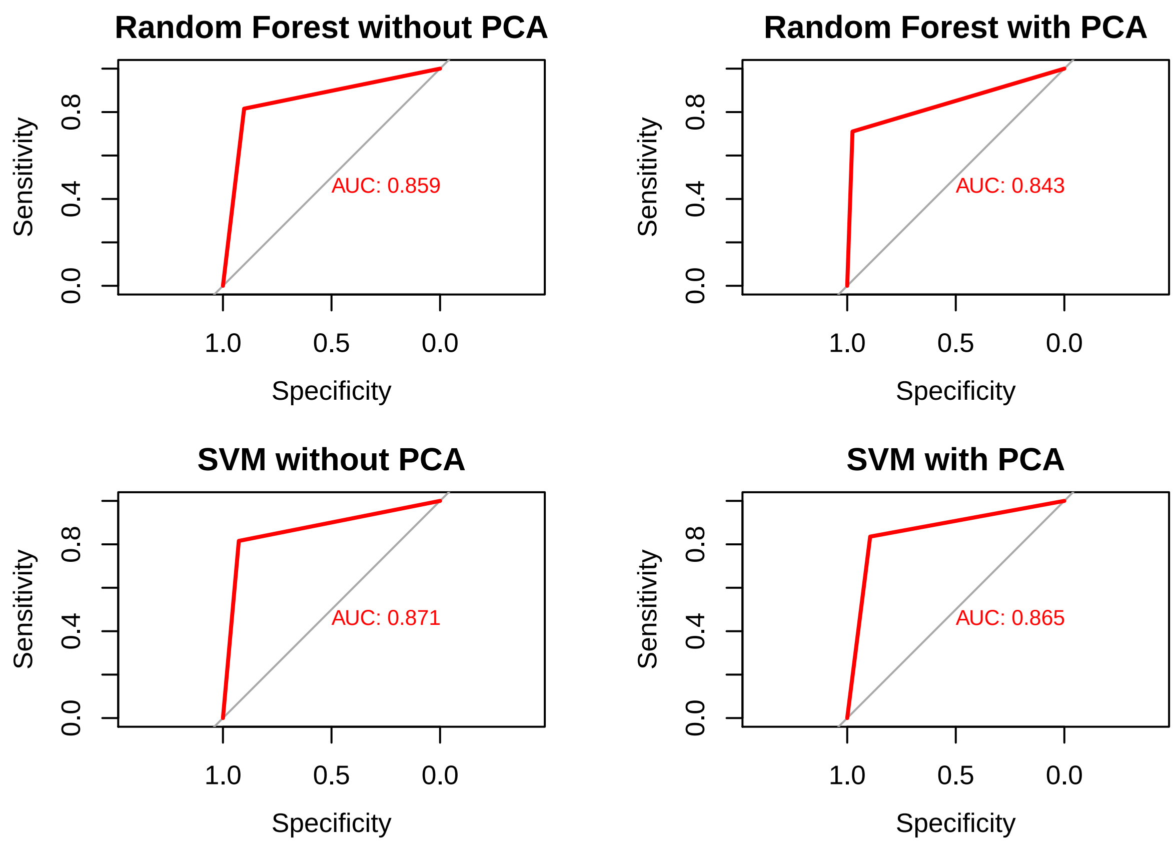
```
# Combine all models' accuracy into rows
acr <- rbind("Random Forest without PCA" = format(round(cm_rf$overall[['Accuracy']], 4),
                                                  nsmall=4),
             "Random Forest with PCA" = format(round(cm_rf_pca$overall[['Accuracy']], 4),
                                               nsmall=4),
             "SVM without PCA" = format(round(cm_svm$overall[['Accuracy']], 4),
                                        nsmall=4),
             "SVM with PCA" = format(round(cm_svm_pca$overall[['Accuracy']], 4),
                                     nsmall=4))

# Update column name and create a html format table
colnames(acr) <- "Accuracy"
knitr::kable(acr)
```

	Accuracy
Random Forest without PCA	0.8545
Random Forest with PCA	0.8291
SVM without PCA	0.8655
SVM with PCA	0.8618

ROC Curves

Below is the ROC curves for both models.

```
# ROC Curves
par(mfrow=c(2,2))
roc_rf <- roc(response = test_data$HeartDisease,
              predictor = as.numeric(pred_rf_mod))
plot(roc_rf,
     col = "red",
     print.auc = TRUE,
     main = 'Random Forest without PCA')
roc_rf_pca <- roc(response = test_data_pca$HeartDisease,
                  predictor = as.numeric(pred_rf_mod_pca))
plot(roc_rf_pca,
     col = "red",
     print.auc = TRUE,
     main = 'Random Forest with PCA')
roc_svm <- roc(response = test_data$HeartDisease,
               predictor = as.numeric(pred_svm_mod))
plot(roc_svm,
     col = "red",
     print.auc = TRUE,
     main = 'SVM without PCA')
roc_svm_pca <- roc(response = test_data_pca$HeartDisease,
                   predictor = as.numeric(pred_svm_mod_pca))
plot(roc_svm_pca,
     col = "red",
     print.auc = TRUE,
     main = 'SVM with PCA')
```



We can notice from the ROC curves which AUC values of support vector machine model trained without PCs data set has higher value compared to the model built and trained with principal components or random forest models built and trained without PCs.

Conclusion

To conclude, support vector machine model trained without PCA data set has higher accuracy in classification to predict heart disease compared to all other 3 models. Dimensionality reduction through principal component analysis does not help to improve classification accuracy through removing the irrelevant, noisy and redundant features in this case but reduce the amount of time to train models. I would recommend to always consider to include PCA as part of a study especially dealing with high-dimensional healthcare dataset even though it may not help to make the model perform better but speeding up the computation with little sacrifice of accuracy is a better solution in many real world problems.

References

1. Centers for Disease Control and Prevention. (2020, September 8). Heart Failure. Retrieved from https://www.cdc.gov/heartdisease/heart_failure.htm#:~:text=Facts%20About%20Heart%20Failure%20in,United%20States

2. Kim H. (2021, June 30). Epidemiology of cardiovascular disease and its risk factors in Korea. National Library of Medicine. Retrieved from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8239378/

3. Sitta Y. (2020, April 13). Time Efficiency and Accuracy Improvement Using PCA. Algoritma Technical Blog. Retrieved from https://algotech.netlify.app/blog/time-and-accuracy-improvement-using-pca/