Capstone - Heart Disease Project

Michael Twiston Davies

02/01/2020

Introduction

This project intends to apply machine learning techniques that go beyond standard linear regression to try and predict the presence of heart disease in patients.

History and disclosure

The data was collected from the four following locations:

1. Cleveland Clinic Foundation (cleveland.data)
2. Hungarian Institute of Cardiology, Budapest (hungarian.data)
3. V.A. Medical Center, Long Beach, CA (long-beach-va.data)
4. University Hospital, Zurich, Switzerland (switzerland.data)

The authors of the databases have requested:

That any publications resulting from the use of the data include the names of the principal investigator responsible for the data collection at each institution. They would be:

1. Hungarian Institute of Cardiology. Budapest: Andras Janosi, M.D.
2. University Hospital, Zurich, Switzerland: William Steinbrunn, M.D.
3. University Hospital, Basel, Switzerland: Matthias Pfisterer, M.D.
4. V.A. Medical Center, Long Beach and Cleveland Clinic Foundation: Robert Detrano, M.D., Ph.D.

Previous results

Robert Detrano, M.D. published his findings on "International application of a new probability algorithm for the diagnosis of coronary artery disease" in the American Journal of Cardiology. He obtained approximately a 77% correct classification accuracy with a logistic-regression-derived discriminant function.

David W. Aha & Dennis Kibler's published work on "Instance-based prediction of heart-disease presence" with the Cleveland database yielded 74.8% to 77.0% accurary.

John Gennari's "Models of incremental concept formation" published in "Artificial Intelligence" utilised a CLASSIT conceptual clustering system achieving 78.9% accuracy on the Cleveland database.

The data set

The full data set can be found here. https://archive.ics.uci.edu/ml/datasets/Heart+Disease

The full database contains 76 attributes, however the 14 listed below are the subset utlised by all published experiments. The Cleveland database is the only one that has been used by machine learning researchers according to archive.ics.uci.edu.

Attribute Information:

Only 14 attributes used (# for original attribute numbering):

Predictors

1. #3 (age) - age in years
2. #4 (sex) - sex (1 = male; 0 = female)
3. #9 (cp) - chest pain type; 1: typical angina, 2: atypical angina, 3: non-anginal pain and 4: asymptomatic
4. #10 (trestbps) - resting blood pressure (in mm Hg on admission to the hospital)
5. #12 (chol) - serum cholestoral in mg/dl
6. #16 (fbs) - (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
7. #19 (restecg) - resting electrocardiographic results; 0: normal, 1: having ST-T wave abnormality, 2: showing probable or definite left ventricular hypertrophy by Estes' criteria
8. #32 (thalach) - maximum heart rate achieved
9. #38 (exang) - exercise induced angina (1 = yes; 0 = no)
10. #40 (oldpeak) - ST depression induced by exercise relative to rest
11. #41 (slope) - the slope of the peak exercise ST segment; 1: upsloping, 2: flat and 3: downsloping
12. #44 (ca) - number of major vessels (0-3) colored by flourosopy
13. #51 (thal) - Form of thalassemia; 3 = normal; 6 = fixed defect; 7 = reversable defect

Predicting

14. #58 (num) - diagnosis of heart disease (angiographic disease status); The presence of heart disease is on a scale 0 being the absence and 1-4 showing presence.

We will visualise the data within the method section.

Aim

We aim to use machine learning to try and create a predictive method to detect heart disease in patients based on predictors.

The presence of heart disease is on a scale, 0 being the absence and 1-4 showing presence. For the purpose of this investigation we are interested in identifying only the presence of heart disease (1-4) and therefore shall allocate these all the value 1 so we can perform logistic regression with a binary outcome.

John Gennari achieved a 78.9% accuracy using his CLASSIT conceptual clustering system on the Cleveland database, we will aim to improve upon this.

Therefore we aim for an accuracy of > 0.789, although we are also interested in the sensitivity and specificity (proportion of correctly identified absence of heart disease and proportion of its presence).

Method

Method - Overview

Our data set will firstly be split 10% into a validation data set for which we will assess the final performance of our algorithm, the remaining 90% will be further split into 80% for training the data and 20% for testing.

We will utilise the below machine learning techniques training these on the training data set and then testing the accuracy on the testing data. We will use these values to pick our final model which we will then validate against the validation data set and try to obtain an accuracy greater than 78.9%. We will also create an ensemble of the best models to increase our accuracy (by using a majority prediction across the models for each outcome).

1. Model 1 - K-Means
2. Model 2 - Partial Least Squares ("PLS")
3. Model 3 - General Logistic Regression ("GLM")
4. Model 4 - Linear Discriminant Analysis ("LDA")
5. Model 5 - Quadratic Discriminant Analysis ("QDA")
6. Model 6 - Local weighted regression ("Loess")
7. Model 7 - K-Nearest Neighbors ("KNN")
8. Model 8 - Random Forest ("RF")

Data Preparation

Load Required Packages

```
############################################################
# Load Packages
############################################################

# packages to be used
library(tidyverse) # a multitude of useful functions
library(caret) # for prediction formulas
library(ggplot2) # graphic visualisations
library(ggcorrplot) # Correlation plot
library(cluster)    # clustering algorithms
library(factoextra) # clustering algorithms & visualization
library(clusterSim) # normalise data for PCA
library(gridExtra) # grid arrange graphs
library(gam) # for Loess models
library(matrixStats) # colsds for scaling

# Suppress summarise info
library(dplyr, warn.conflicts = FALSE) # use to suppress grouping warning messages
options(dplyr.summarise.inform = FALSE) # use to suppress grouping warning messages
```

Pull data set from ics.uci.edu

Below we will pull the data from https://archive.ics.uci.edu/ml/datasets/Heart+Disease. However, the layout is in one column with spaces splitting the fourteen data points for each observation. We will address this and rename the columns based on the names given to us from the webpage's "Attibute information" detailed above. Then we will add a fifteenth column to give each data set a location as this might prove useful later on.

This directory contains 4 databases concerning heart disease diagnosis which we will pull individually. All attributes are numeric-valued. The data was collected from the four following locations:

1. Cleveland Clinic Foundation (cleveland.data)
2. Hungarian Institute of Cardiology, Budapest (hungarian.data)
3. V.A. Medical Center, Long Beach, CA (long-beach-va.data)
4. University Hospital, Zurich, Switzerland (switzerland.data)

```
############################################################
# Pull data set from ics.uci.edu
############################################################

# reprocessed.hungarian.data
heart.uci.hungarian <- read.csv(url("https://archive.ics.uci.edu/ml/machine-learning-databases/heart-dis
                        header=FALSE) %>%
  separate(col = V1 ,
           sep = "\\s",
           into = c("age", "sex", "cp", "trestbps", "chol", "fbs", "restecg", "thalach", "exang", "oldpe
  mutate(Location = c("Hungary"))

# processed.cleveland.data
heart.uci.cleveland <- read.csv(url("https://archive.ics.uci.edu/ml/machine-learning-databases/heart-dis
                                header=FALSE)

colnames(heart.uci.cleveland) <- c("age", "sex", "cp", "trestbps", "chol", "fbs", "restecg", "thalach",

heart.uci.cleveland <- heart.uci.cleveland %>%
  mutate(Location = c("Cleveland"))

# processed.switzerland.data
heart.uci.switzerland <- read.csv(url("https://archive.ics.uci.edu/ml/machine-learning-databases/heart-c
                                header=FALSE)

colnames(heart.uci.switzerland) <- c("age", "sex", "cp", "trestbps", "chol", "fbs", "restecg", "thalach

heart.uci.switzerland <- heart.uci.switzerland %>%
  mutate(Location = c("Switzerland"))


# processed.va.data
heart.uci.longbeach <- read.csv(url("https://archive.ics.uci.edu/ml/machine-learning-databases/heart-dis
                                header=FALSE)

colnames(heart.uci.longbeach) <- c("age", "sex", "cp", "trestbps", "chol", "fbs", "restecg", "thalach",

heart.uci.longbeach <- heart.uci.longbeach %>%
  mutate(Location = c("LongBeach"))
```

We note that some of this data needs cleaning. Most obvious is the use of "?" for NA values. We will replace these soon.

Though we will do this once for the final data set so firstly we will append them together.

```
heart.uci <- rbind( heart.uci.cleveland , heart.uci.hungarian , heart.uci.longbeach , heart.uci.switzerl

# Now we can remove the starting four data sets.
rm( heart.uci.cleveland , heart.uci.hungarian , heart.uci.longbeach , heart.uci.switzerland)
```

Clean the data

In this part we will remove rows without a outcome logged and those with symbol values or incorrect values will be replaced.

```
############################################################
# Data Cleaning
############################################################
```

There seems to be a row with blank values. Since we are predicting the "num" variable we will filter this out from this column.

```
unique(heart.uci$num) # view unique values in outcome column to show this is not complete
```

```
## [1] "0" "2" "1" "3" "4" ""
```

```
heart.uci <- heart.uci %>%
  filter( num != "") #remove blank values
```

Though we still have the "?" values which we will replace with NA as we will also filter these out with na.omit() later.

```
heart.uci <- heart.uci %>%
  na_if("?")
```

The information provided with the data set also states that a number of attribute values are missing which is distinguished by a "-9.0" value. We will therefore also replace these with NA across the whole data set.

```
heart.uci <- heart.uci %>%
  na_if("-9.0") %>%
  na_if("-9") %>%
  na_if(-9)
```

We also note that the "thal" and "ca" columns have values which are the same numerically but different strings, we replace these below.

```
heart.uci <- heart.uci %>%
  mutate(thal = if_else(thal == "7", "7.0",thal)) %>%
  mutate(ca = if_else(ca == "0", "0.0",ca))
```

Visualising the data by location

We tagged the 4 sets of data by location, here we will see if we can increase the data set used from previous studies which only utilised the Cleveland data.

```
##########

# Quick look at the data by location - All data
heart.uci.location <- heart.uci %>%
  gather(key = "variable", value = "value", -one_of("Location")) %>%
  group_by(Location , variable) %>%
  summarise(n = n())

heart.uci.location.summary <- heart.uci.location %>%
  spread(key = "variable", value = "n")
```

The below table shows us that there is potentially another ~617 in addition to the 303 records from Cleveland.

```
heart.uci.location.summary %>% knitr::kable()
```

Location	age	ca	chol	cp	exang	fbs	num	oldpeak	restecg	sex	slope	thal	thalach	trestbps
Cleveland	303	303	303	303	303	303	303	303	303	303	303	303	303	303
Hungary	294	294	294	294	294	294	294	294	294	294	294	294	294	294
LongBeach	200	200	200	200	200	200	200	200	200	200	200	200	200	200
Switzerland	123	123	123	123	123	123	123	123	123	123	123	123	123	123

However, when removing "NA" values (in the table below) we can see that "ca" and "thal" are not recorded in locations other than Cleveland regularly. If we removed these variables and then filtered out all NA values we would end up with 531 rows of observations. This would be a dramatic increase on our data set from ~299, however we will later discover that via the "random forest" method that these are important variables.

Therefore we will not remove these variables to increase the number of observations. I ran the models excluding "ca" and "thal" which decreased their accuracy, justifying this choice.

```
# Quick look at the data by location - Data excluding NA's

heart.uci.location <- heart.uci %>%
  gather(key = "variable", value = "value", -one_of("Location")) %>%
  filter(!is.na(value)) %>%
  group_by(Location , variable) %>%
  summarise(n = n())

heart.uci.location.summary <- heart.uci.location %>%
  spread(key = "variable", value = "n")

heart.uci.location.summary %>% knitr::kable()
```

Location	age	ca	chol	cp	exang	fbs	num	oldpeak	restecg	sex	slope	thal	thalach	trestbps
Cleveland	303	299	303	303	303	303	303	303	303	303	303	301	303	303
Hungary	294	4	271	294	293	286	294	294	293	294	104	28	293	293

Location	age	ca	chol	cp	exang	fbs	num	oldpeak	restecg	sex	slope	thal	thalach	trestbps
LongBeach	200	2	193	200	147	193	200	144	200	200	98	34	147	144
Switzerland	123	5	123	123	122	48	123	117	122	123	106	71	122	121

```
##########

# Remove NA values
heart.uci <- na.omit(heart.uci)

# Number of observations by location
heart.uci.location <- heart.uci %>%
  gather(key = "variable", value = "value", -one_of("Location")) %>%
  filter( variable == "num") %>%
  group_by(Location , variable) %>%
  summarise(n = n())

rm(heart.uci.location.summary , heart.uci.location)
```

After deciding to keep "ca" and "thal" values whilst also removing all rows with NA data points we can see that 297 are relating to the Cleveland data, 1 to Hungary and 1 LongBeach. It therefore seems pointless to keep the location variable so we will remove it.

```
heart.uci <- heart.uci %>%
  dplyr::select(-Location) %>%   # Remove location variable added.
  mutate( PresenceValue = if_else(num == 0 , "0" , "1") )  # replace "num" with "PresencValue" as we ar
```

Store data for visualisation purposes

Before we split our data into validation, test and training partitions lets just copy the data to be used for our visualisation section where we will select predictors to be used from the variables available.

```
# Data for visualisation. We are copying the database for the exploration phase before our machine lear
heart.visualise <- heart.uci
```

Split data set into testing, training and validation data sets

As mentioned in the introduction we will now split out the validation, test and training sets set.

```
############################################################
# Split data set into testing, training and validation data sets (final hold-out test set)
############################################################

# Remove num
heart.uci <- heart.uci %>%
  dplyr::select( - num)

# Change columns to numeric initially for analysis. We will transform those with levels back into facto
# Columns to be changed
cols <- c("age", "sex", "cp", "trestbps", "chol", "fbs", "restecg","thalach", "exang","oldpeak", "slope
# Apply change
```

```
heart.uci[cols] <- lapply(heart.uci[cols], as.numeric)

# Columns to be changed back into factors
cols <- c("sex", "cp", "fbs", "restecg", "exang", "slope", "thal", "PresenceValue")

# Apply change
heart.uci[cols] <- lapply(heart.uci[cols], factor)
rm(cols)

# Validation set will be 10% of the data. We will use this to test our final model.

set.seed(1) # Set seed to ensure those repeating this analysis obtain the same results.
test_index <- createDataPartition(y = heart.uci$PresenceValue, times = 1, p = 0.1, list = FALSE)
heartdata <- heart.uci[-test_index,]
validation <- heart.uci[test_index,]

# Remove items no longer needed.
rm(test_index)
```

We converted the columns to numeric and then those with levels back into factors before splitting the data in order to ensure we have observations at each level for all factors in the split data sets. We can see this is true with the below summary tables.

```
summary(heartdata) %>% knitr::kable()
```

age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	PresenceValue
Min. :29.00	0: 90	1: 19	Min. : 94.0	Min. :100.0	0:230	0:136	Min. : 71.0	0:180	Min. :0.000	1:130	Min. :0.000	3:148	0:144
1st Qu.:47.00	1:179	2: 46	1st Qu.:120.0	1st Qu.:211.0	1: 39	1: 3	1st Qu.:132.0	1: 89	1st Qu.:0.000	2:120	1st Qu.:0.000	6: 16	1:125
Median :56.00	NA	3: 75	Median :130.0	Median :241.0	NA	2:130	Median :153.0	NA	Median :0.800	3: 19	Median :0.000	7:105	NA
Mean :54.28	NA	4:129	Mean :132.1	Mean :245.5	NA	NA	Mean :149.3	NA	Mean :1.057	NA	Mean :0.632	NA	NA
3rd Qu.:61.00	NA	NA	3rd Qu.:140.0	3rd Qu.:275.0	NA	NA	3rd Qu.:166.0	NA	3rd Qu.:1.600	NA	3rd Qu.:1.000	NA	NA
Max. :76.00	NA	NA	Max. :200.0	Max. :417.0	NA	NA	Max. :202.0	NA	Max. :6.200	NA	Max. :3.000	NA	NA

```
summary(validation) %>% knitr::kable()
```

age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	PresenceValue
Min. :43.00	0: 6	1: 4	Min. :100.0	Min. :126.0	0:26	0:13	Min. :109.0	0:20	Min. :0.000	1: 9	Min. :0.000	3:16	0:16
1st Qu.:50.25	1:24	2: 3	1st Qu.:116.2	1st Qu.:214.8	1: 4	1: 1	1st Qu.:139.2	1:10	1st Qu.:0.200	2:19	1st Qu.:0.000	6: 2	1:14
Median :56.50	NA	3: 8	Median :125.0	Median :244.5	NA	2:16	Median :152.0	NA	Median :0.700	3: 2	Median :0.500	7:12	NA
Mean :56.73	NA	4:15	Mean :128.1	Mean :258.7	NA	NA	Mean :149.6	NA	Mean :1.070	NA	Mean :1.033	NA	NA

age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	PresenceValue
3rd Qu.:65.50	NA	NA	3rd Qu.:139.5	3rd Qu.:297.0	NA	NA	3rd Qu.:162.8	NA	3rd Qu.:1.525	NA	3rd Qu.:2.000	NA	NA
Max. :77.00	NA	NA	Max. :160.0	Max. :564.0	NA	NA	Max. :186.0	NA	Max. :4.400	NA	Max. :3.000	NA	NA

```
############################################################
# Partition heartdata data set into test and training data sets
############################################################

set.seed(1) # Set seed to ensure those repeating this analysis obtain the same results.
test_index <- createDataPartition(y = heartdata$PresenceValue, times = 1,
                                  p = 0.2, list = FALSE)

train_set <- heartdata[-test_index,]
test_set <- heartdata[test_index,]
rm(test_index , heartdata)

summary(train_set) %>% knitr::kable()
```

age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	PresenceValue
Min. :29.00	0: 71	1: 18	Min. : 94.0	Min. :141.0	0:184	0:111	Min. : 71.0	0:144	Min. :0.000	1:102	Min. :0.0000	3:122	0:115
1st Qu.:47.50	1:144	2: 39	1st Qu.:120.0	1st Qu.:209.0	1: 31	1: 3	1st Qu.:132.0	1: 71	1st Qu.:0.000	2: 95	1st Qu.:0.0000	6: 13	1:100
Median :56.00	NA	3: 58	Median :130.0	Median :239.0	NA	2:101	Median :155.0	NA	Median :0.800	3: 18	Median :0.0000	7: 80	NA
Mean :54.19	NA	4:100	Mean :131.9	Mean :245.6	NA	NA	Mean :149.5	NA	Mean :1.109	NA	Mean :0.6186	NA	NA
3rd Qu.:60.00	NA	NA	3rd Qu.:140.0	3rd Qu.:275.5	NA	NA	3rd Qu.:167.5	NA	3rd Qu.:1.800	NA	3rd Qu.:1.0000	NA	NA
Max. :76.00	NA	NA	Max. :200.0	Max. :417.0	NA	NA	Max. :202.0	NA	Max. :6.200	NA	Max. :3.0000	NA	NA

```
summary(test_set) %>% knitr::kable()
```

age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	PresenceValue
Min. :34.00	0:19	1: 1	Min. :102.0	Min. :100.0	0:46	0:25	Min. : 90.0	0:36	Min. :0.000	1:28	Min. :0.0000	3:26	0:29
1st Qu.:47.25	1:35	2: 7	1st Qu.:120.0	1st Qu.:220.8	1: 8	1: 0	1st Qu.:141.0	1:18	1st Qu.:0.000	2:25	1st Qu.:0.0000	6: 3	1:25
Median :55.00	NA	3:17	Median :130.0	Median :255.5	NA	2:29	Median :150.0	NA	Median :0.450	3: 1	Median :0.0000	7:25	NA
Mean :54.63	NA	4:29	Mean :132.9	Mean :244.9	NA	NA	Mean :148.5	NA	Mean :0.850	NA	Mean :0.6852	NA	NA
3rd Qu.:61.75	NA	NA	3rd Qu.:144.8	3rd Qu.:269.0	NA	NA	3rd Qu.:164.2	NA	3rd Qu.:1.425	NA	3rd Qu.:1.0000	NA	NA
Max. :70.00	NA	NA	Max. :192.0	Max. :360.0	NA	NA	Max. :195.0	NA	Max. :3.400	NA	Max. :3.0000	NA	NA

Adding/mutating columns for data visualisation

```
############################################################
# Adding information / Changing columns for visualisation/data exploration
############################################################

heart.visualise <- heart.visualise %>%
  # The names and social security numbers of the patients were removed. Although we have no way of know
  mutate(id = 1:n()) %>%
  # Add Male/female descriptions - (sex) - (1 = male; 0 = female)
  mutate( SexDesc = if_else(sex == 1 , "male" , "female") ) %>%
  # (cp) - chest pain type; 1: typical angina, 2: atypical angina, 3: non-anginal pain and 4: asymptoma
  mutate( ChestPain =   case_when(cp == 1 ~ "typical angina",
                                  cp == 2 ~ "atypical angina",
                                  cp == 3 ~ "non-anginal pain",
                                  cp == 4 ~ "asymptomatic"
                                  ) ) %>%
  # (fbs) - (fasting blood sugar > 120 mg/dl)  (1 = true; 0 = false)
  mutate( FastingBloodSugar = if_else(fbs == 1 , "true" , "false") ) %>%
  # (restecg) - resting electrocardiographic results; 0: normal, 1: having ST-T wave abnormality, 2: sh
  mutate( RestingECG =   case_when(restecg == 0 ~ "normal",
                                   restecg == 1 ~ "ST-T wave abnormality",
                                   restecg == 2 ~ "left ventricular hypertrophy"
                                  ) ) %>%
  # (exang) - exercise induced angina (1 = yes; 0 = no)
  mutate( ExerciseInducedAngina = if_else(exang == 1 , "true" , "false") ) %>%
  # (slope) - the slope of the peak exercise ST segment; 1: upsloping, 2: flat and 3: downsloping
  mutate( SlopeDesc =   case_when(slope == 1 ~ "upsloping",
                                   slope == 2 ~ "flat",
                                   slope == 3 ~ "downsloping"
                                   ) ) %>%
   # (thal) - 3 = normal; 6 = fixed defect; 7 = reversable defect
  mutate( thalassemia =   case_when(thal == "3.0" ~ "normal",
                                    thal == "6.0" ~ "fixed defect",
                                    thal == "7.0" ~ "reversable defect"
                                    ) ) %>%
  # Add decription for final diagnosis
  mutate( Diagnosis = case_when(num == 0 ~ "healthy" ,
                                num == 1 ~ "presence 1",
                                num == 2 ~ "presence 2",
                                num == 3 ~ "presence 3",
                                num == 4 ~ "presence 4"
                                ) ) %>%
  # Presence of heart disease (added so we can also just distinguish presence rather than severity)
  mutate( Presence = if_else(num == 0 , "healthy" , "presence") )


# The categorical columns are currently in a numeric format and we need to transform these into factors

# Columns to be changed
cols <- c("sex", "cp", "fbs", "restecg", "exang", "slope", "thal", "num", "SexDesc" , "ChestPain" ,
          "FastingBloodSugar" ,"RestingECG" , "ExerciseInducedAngina" , "SlopeDesc" ,"thalassemia" , "D
```

```
# Apply change
heart.visualise[cols] <- lapply(heart.visualise[cols], factor)
rm(cols)
```

Method - Data exploration and predictor selection for models

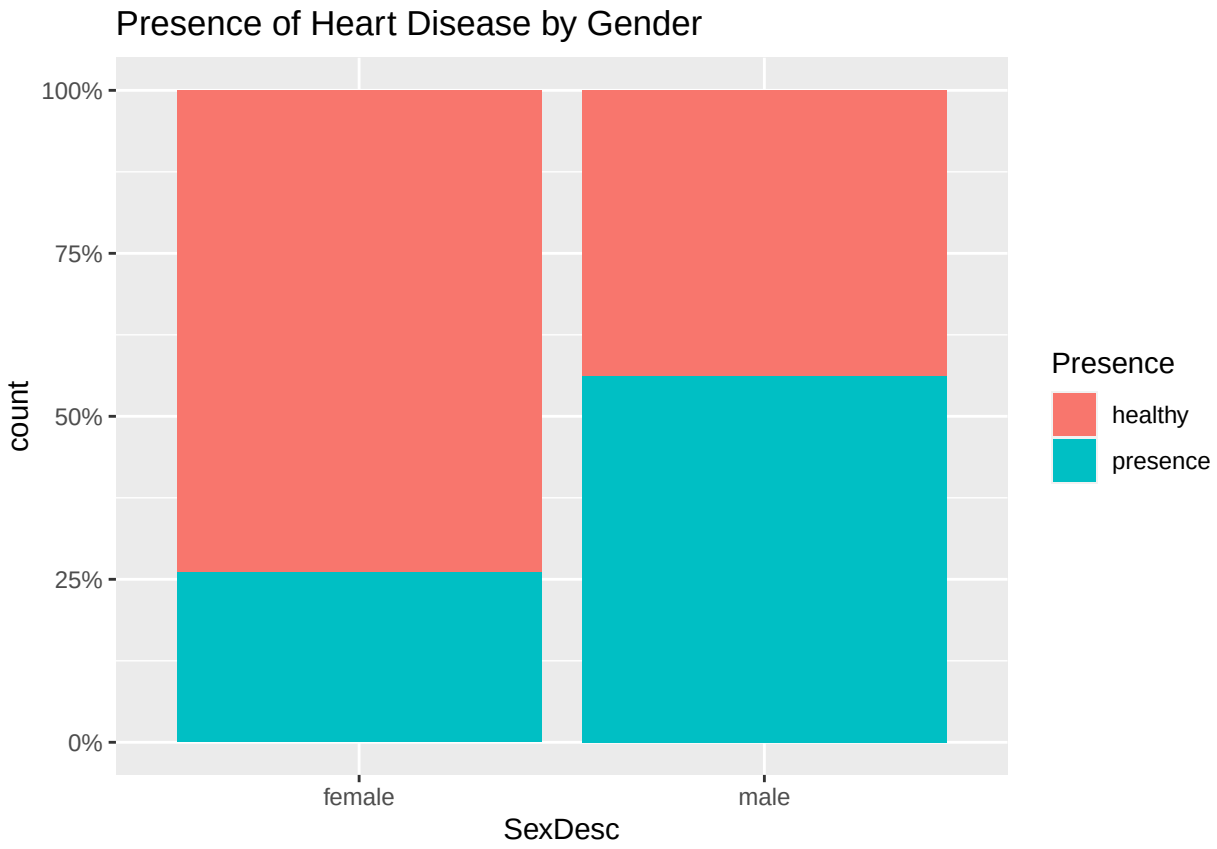
To start us off lets just have a quick look at the first 6 lines in the data set.

```
##   age sex cp trestbps chol fbs restecg thalach exang oldpeak slope  ca thal num
## 1  63   1  1      145  233   1       2     150     0     2.3     3 0.0  6.0   0
## 2  67   1  4      160  286   0       2     108     1     1.5     2 3.0  3.0   2
## 3  67   1  4      120  229   0       2     129     1     2.6     2 2.0  7.0   1
## 4  37   1  3      130  250   0       0     187     0     3.5     3 0.0  3.0   0
## 5  41   0  2      130  204   0       2     172     0     1.4     1 0.0  3.0   0
## 6  56   1  2      120  236   0       0     178     0     0.8     1 0.0  3.0   0
##   PresenceValue id SexDesc        ChestPain FastingBloodSugar
## 1             0  1    male   typical angina              true
## 2             1  2    male     asymptomatic             false
## 3             1  3    male     asymptomatic             false
## 4             0  4    male non-anginal pain             false
## 5             0  5  female  atypical angina             false
## 6             0  6    male  atypical angina             false
##                      RestingECG ExerciseInducedAngina   SlopeDesc
## 1 left ventricular hypertrophy                 false downsloping
## 2 left ventricular hypertrophy                  true        flat
## 3 left ventricular hypertrophy                  true        flat
## 4                       normal                 false downsloping
## 5 left ventricular hypertrophy                 false   upsloping
## 6                       normal                 false   upsloping
##         thalassemia  Diagnosis Presence
## 1      fixed defect    healthy  healthy
## 2            normal presence 2 presence
## 3 reversable defect presence 1 presence
## 4            normal    healthy  healthy
## 5            normal    healthy  healthy
## 6            normal    healthy  healthy
```

```
## [1] "There are 299 observations in the data set"
```

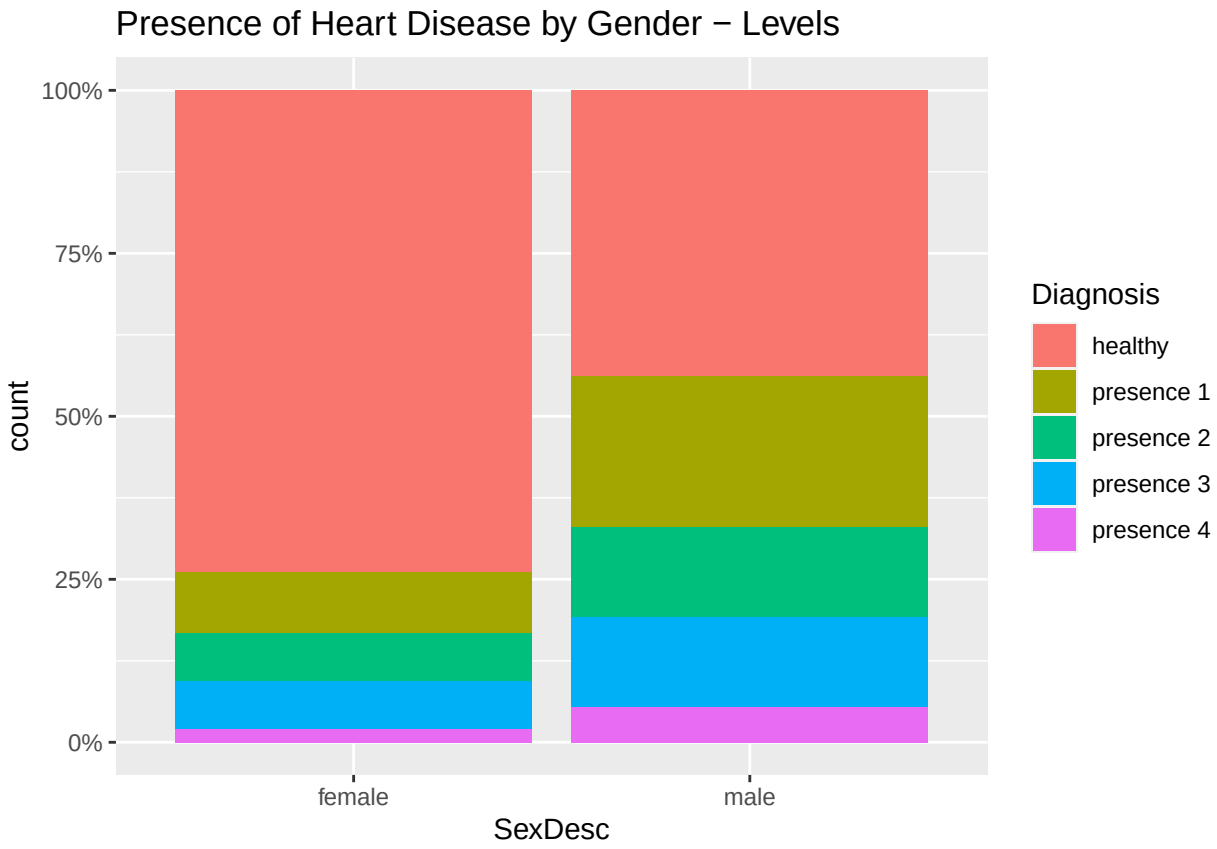
Method - Data exploration - Gender

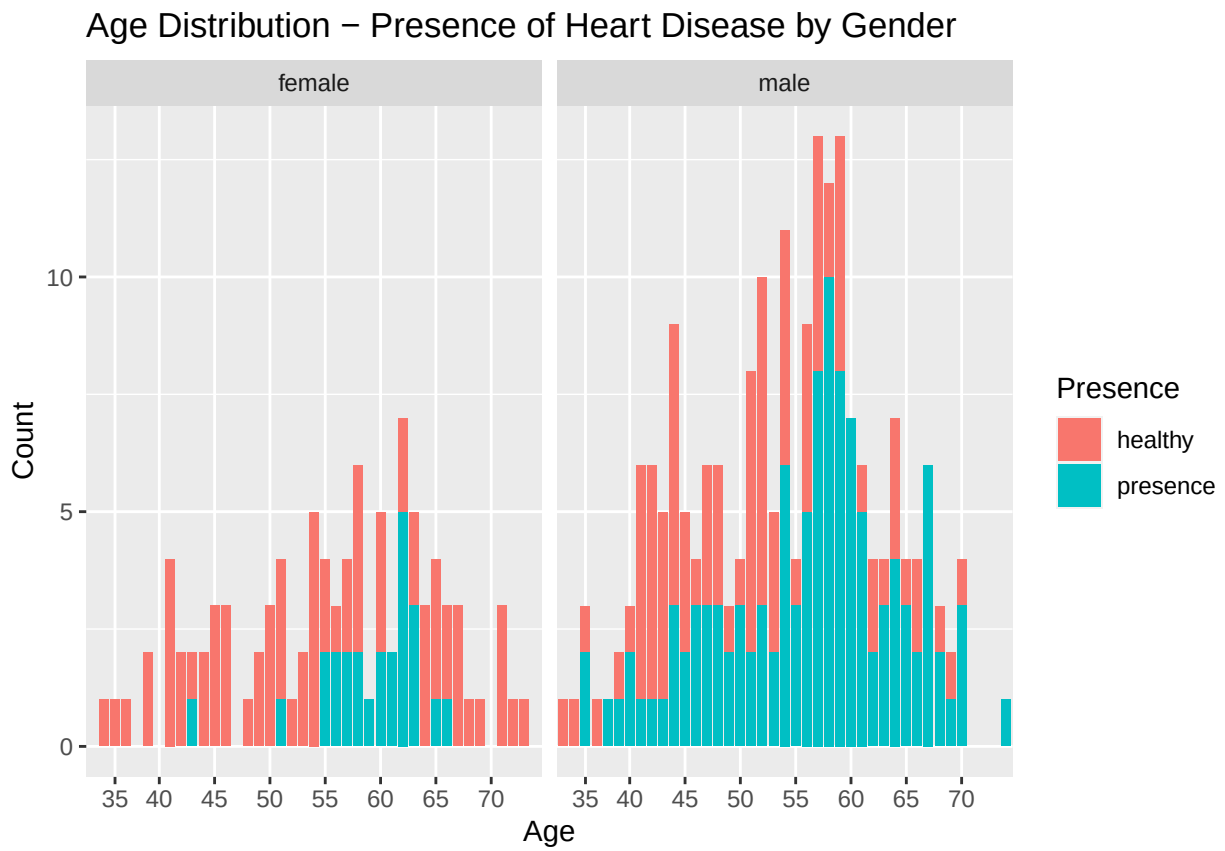
It is likely that we will find a difference between genders so let's take a look at the prevalence between the two.



In our data it seems that heart disease is more prevalent in men, however is this due to sex or other reasons? We need to watch out for causality.

We can see that all Levels within “num” (1-4) form a larger proportion in men.


Presence of Heart Disease by Gender − Levels
100%
75%
50%
25%
0%
count
female
male
SexDesc
Diagnosis
healthy
presence 1
presence 2
presence 3
presence 4



From the above view it could be seen that age is more important than sex as they seems to follow similar distributions, just we have more male cases than female.

Method - Data exploration - Factor Variables

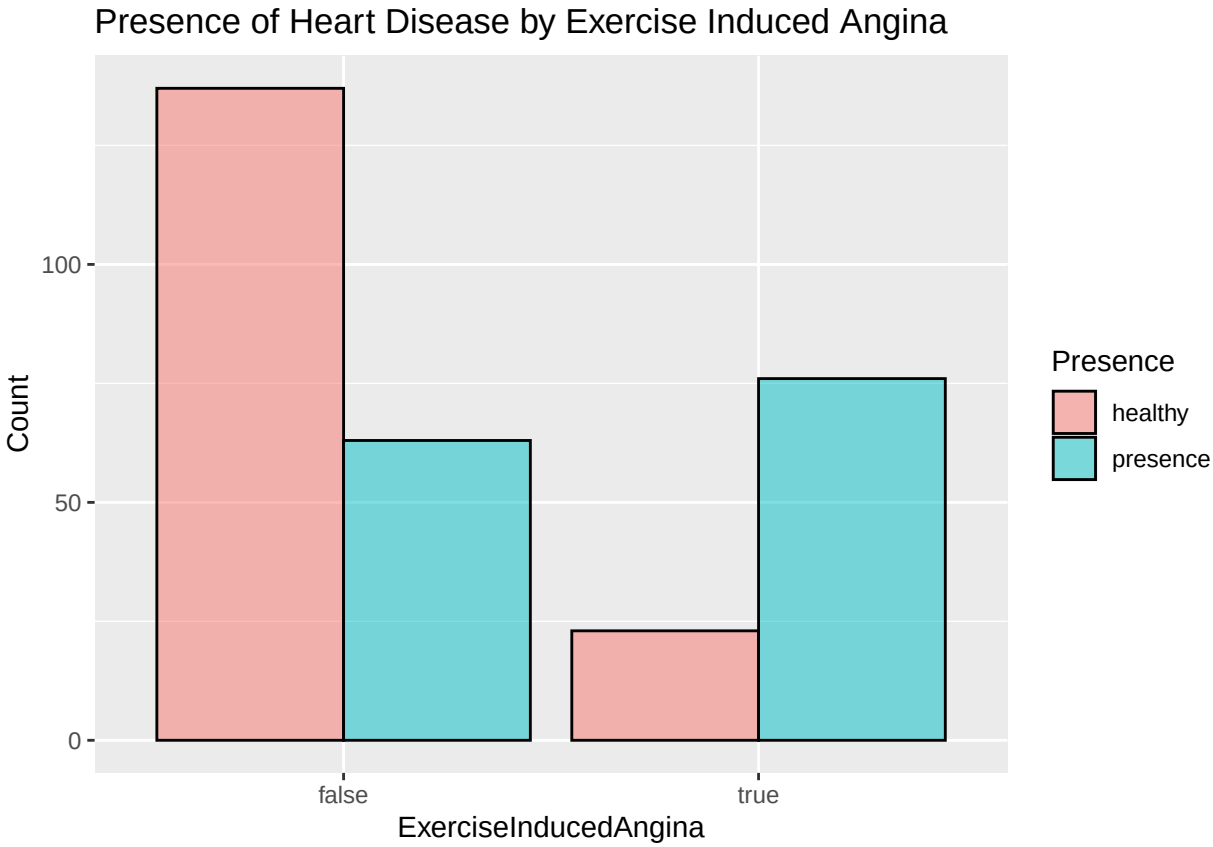
Now lets see if we can identify some other key variables to use as predictors within our models.

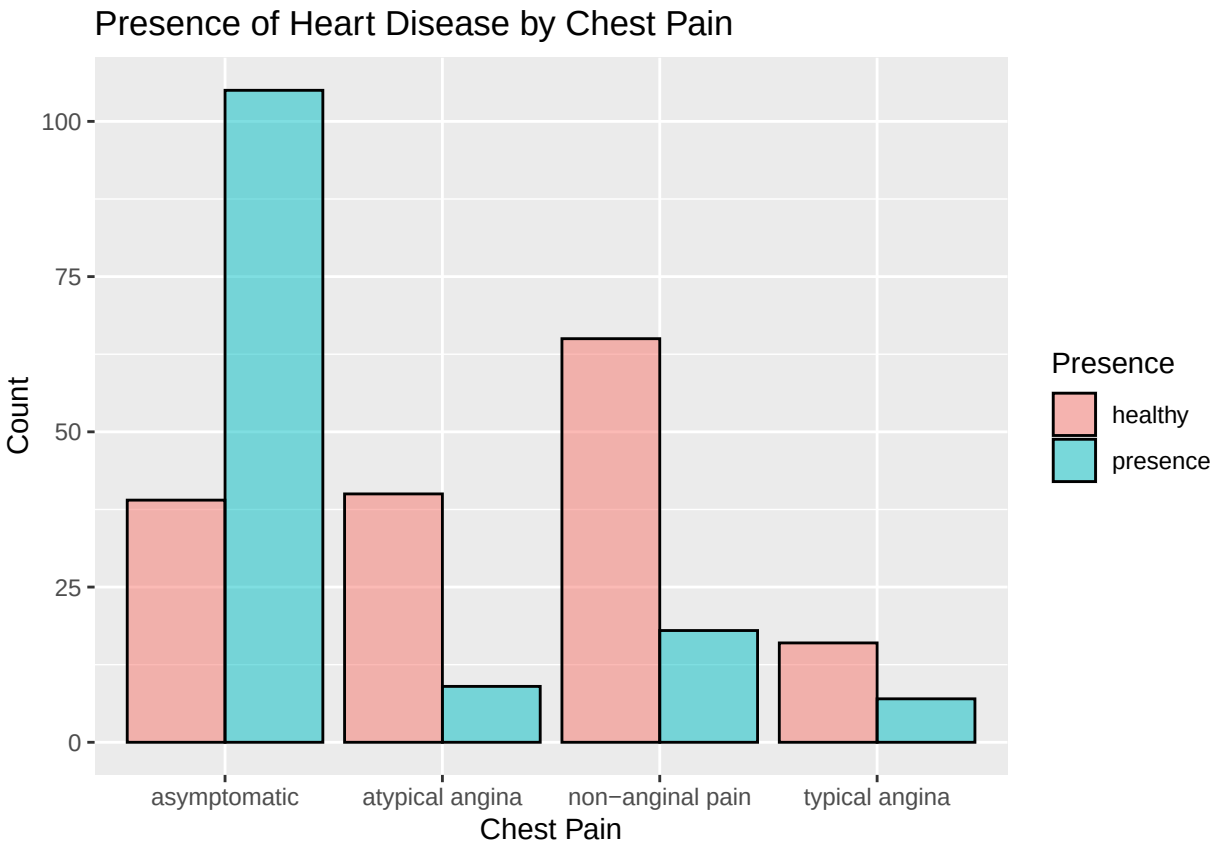
We will do this as using all 13 potential variables to predict the presence of heart disease will likely lead to overfitting.

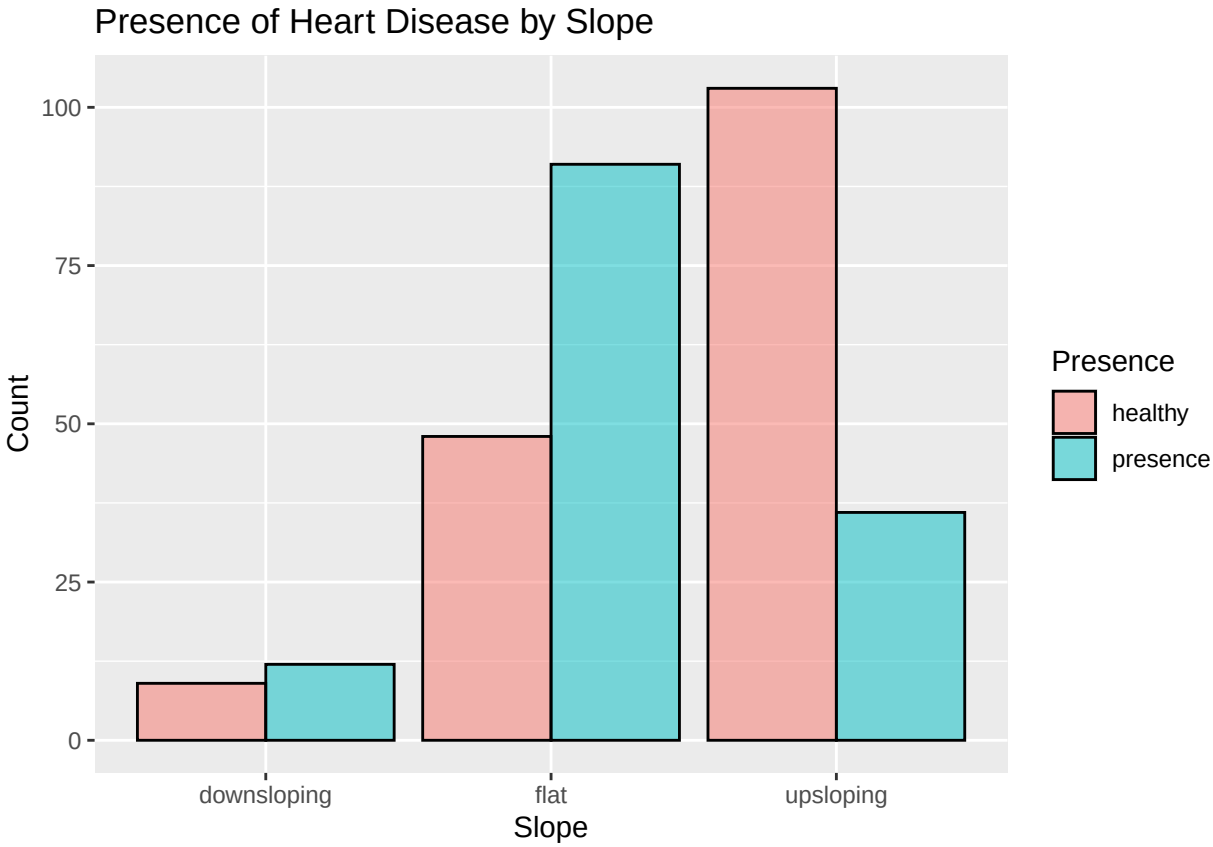
Firstly let's look at the factor variables.

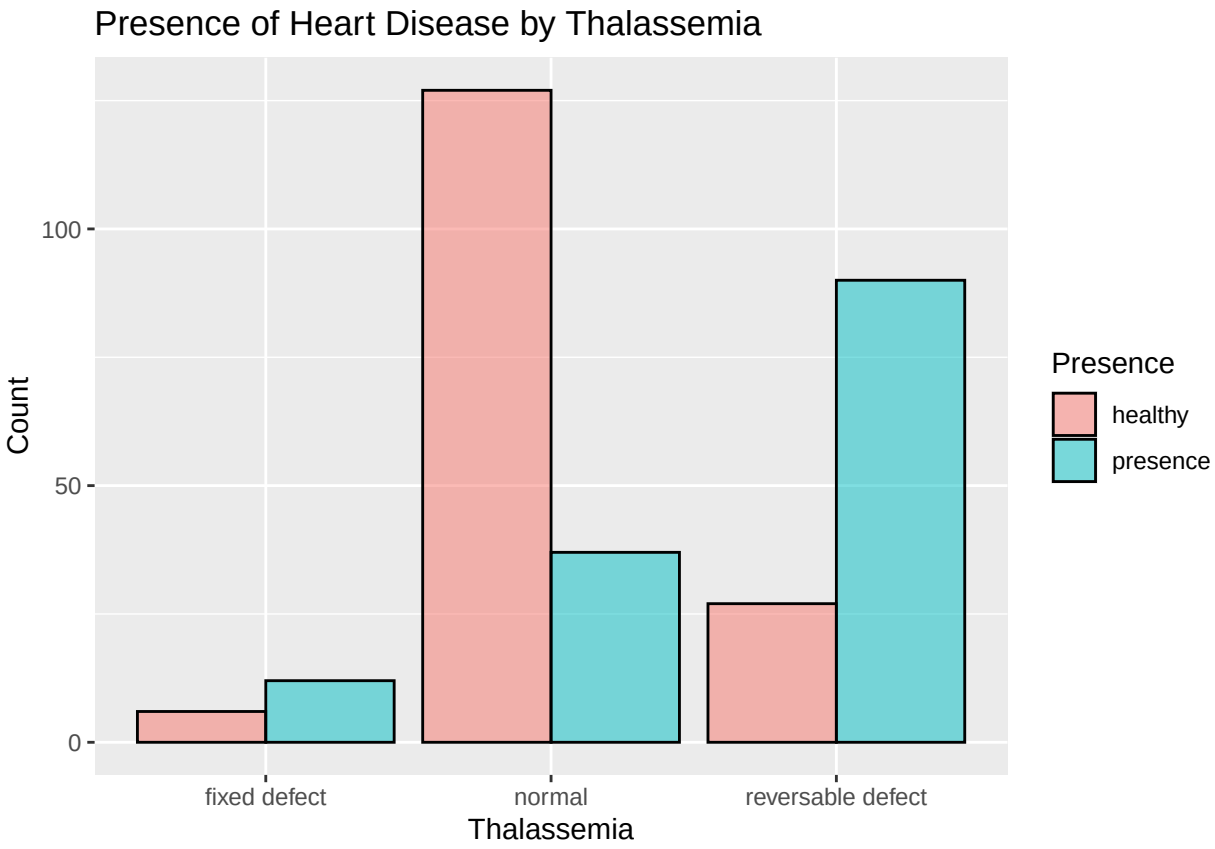
These have been split into good/bad predictors based on the difference in outcomes for each level. The larger the difference at each level the more likely this will improve the accuracy of our models.

Good potential predictors Firstly let's look at some good potential predictors from our factor variables.

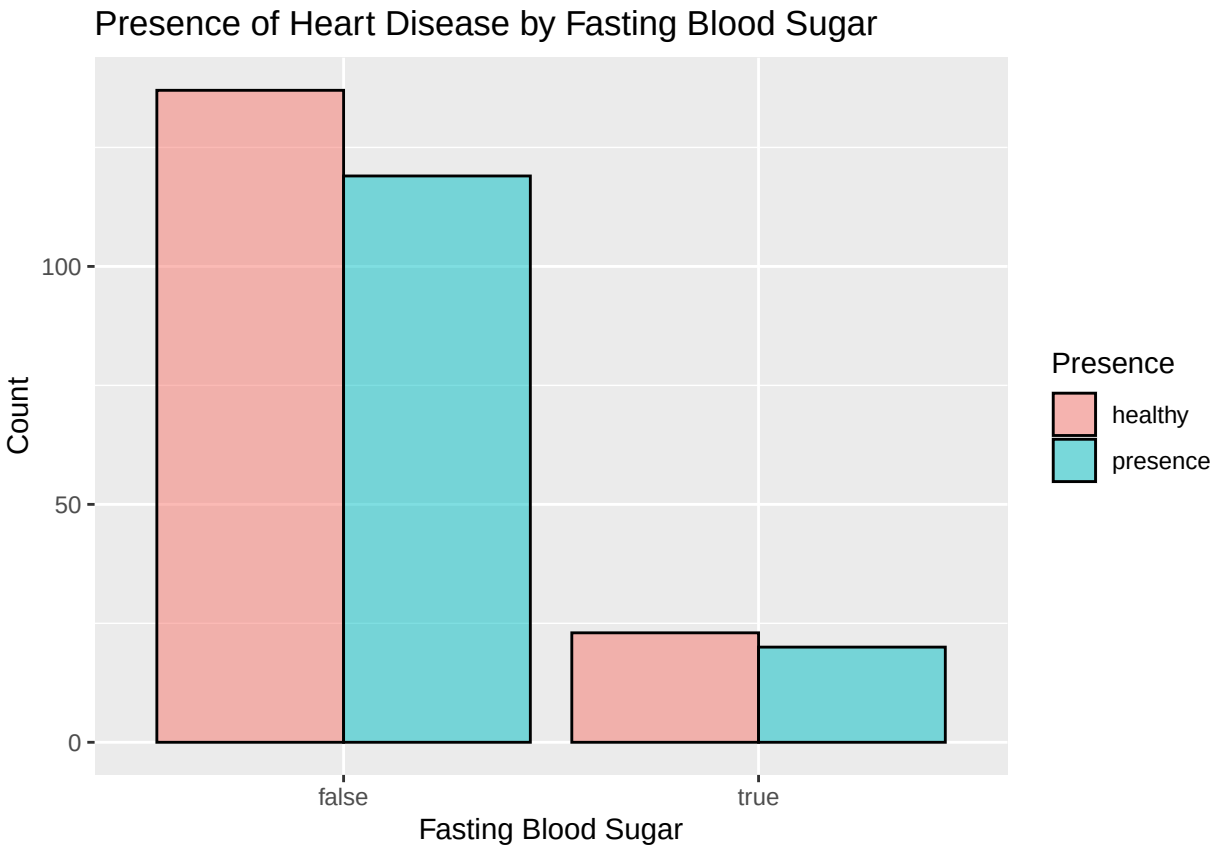

Presence of Heart Disease by Exercise Induced Angina
Count
100
50
0
false
true
ExerciseInducedAngina
Presence
healthy
presence

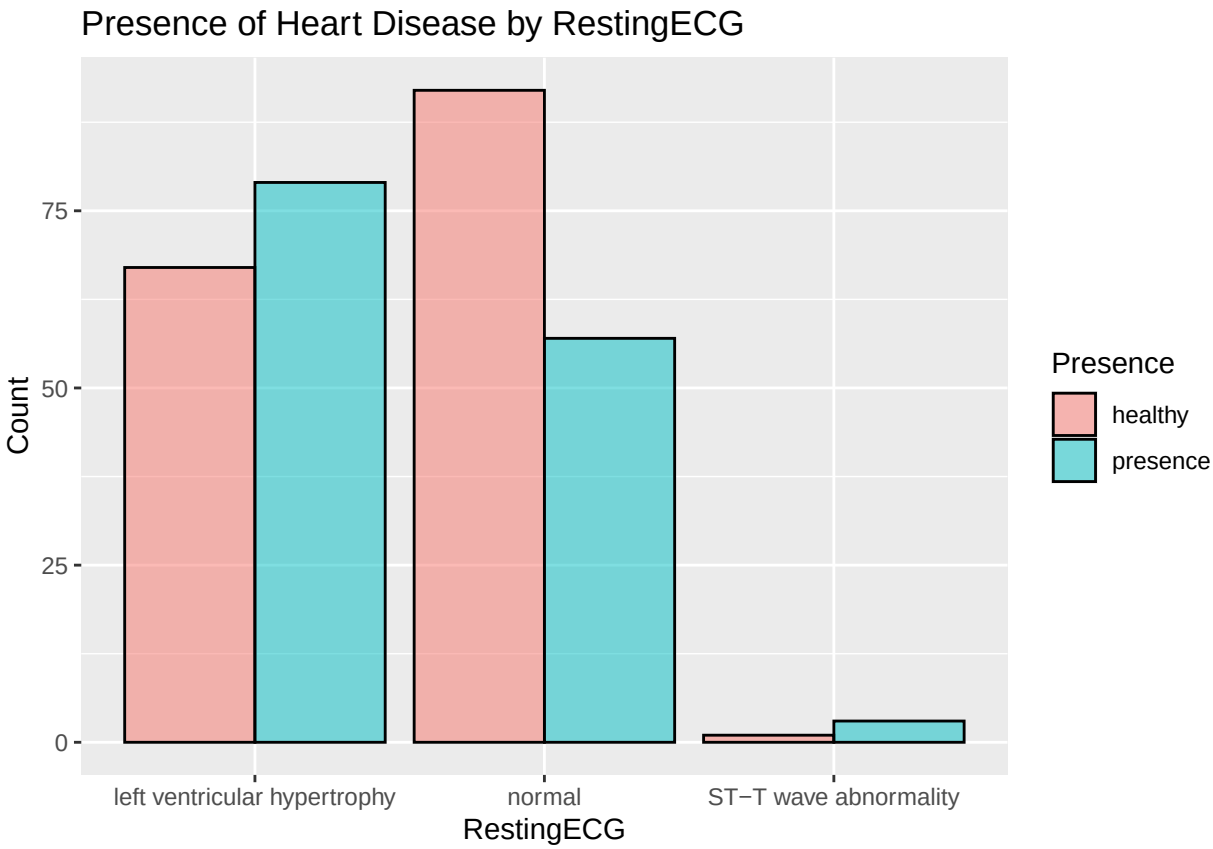

Presence of Heart Disease by Chest Pain
Count
100
75
50
25
0
asymptomatic
atypical angina
non−anginal pain
typical angina
Chest Pain
Presence
healthy
presence

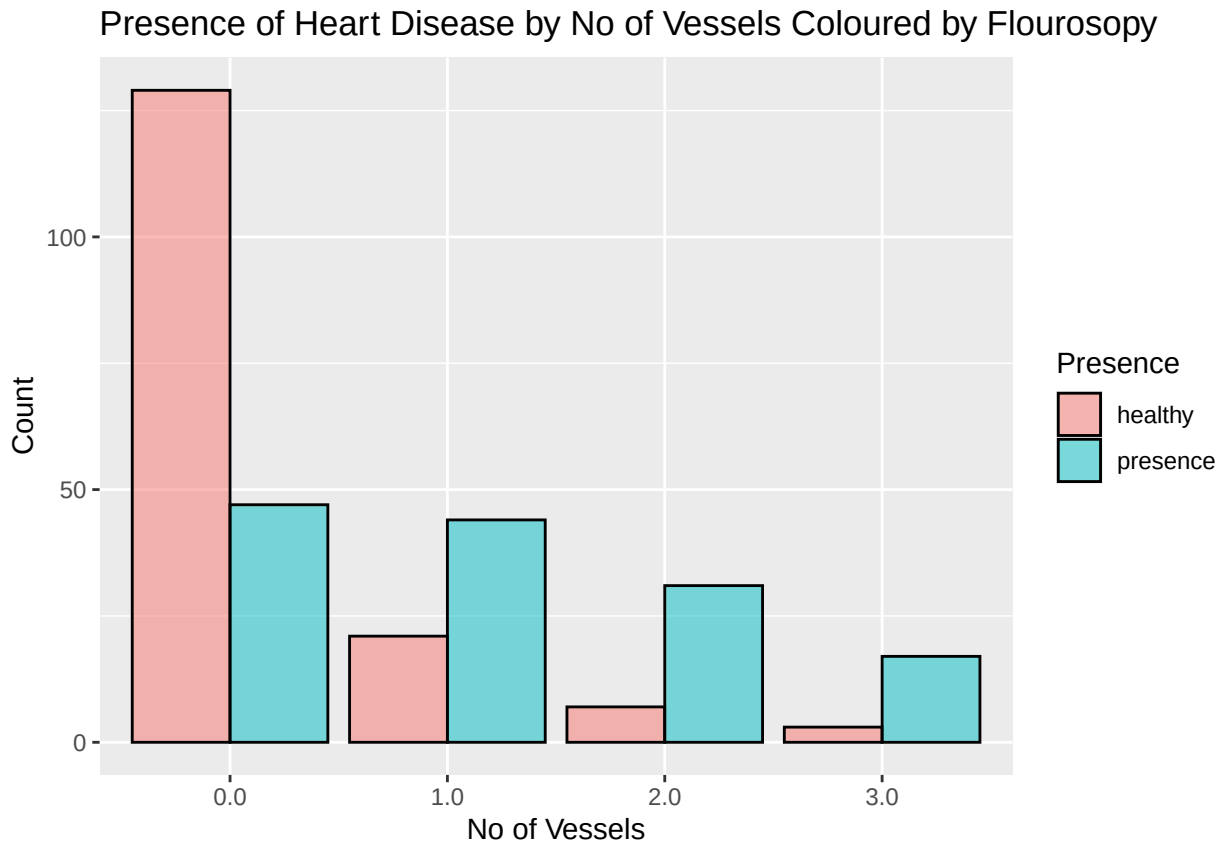

Presence of Heart Disease by Slope
Count
100
75
50
25
0
downsloping
flat
upsloping
Slope
Presence
healthy
presence



Likely less good potential predictors Now some potentially bad predictors from our factor variables.


Presence of Heart Disease by Fasting Blood Sugar
Count
100
50
0
false
true
Fasting Blood Sugar
Presence
healthy
presence

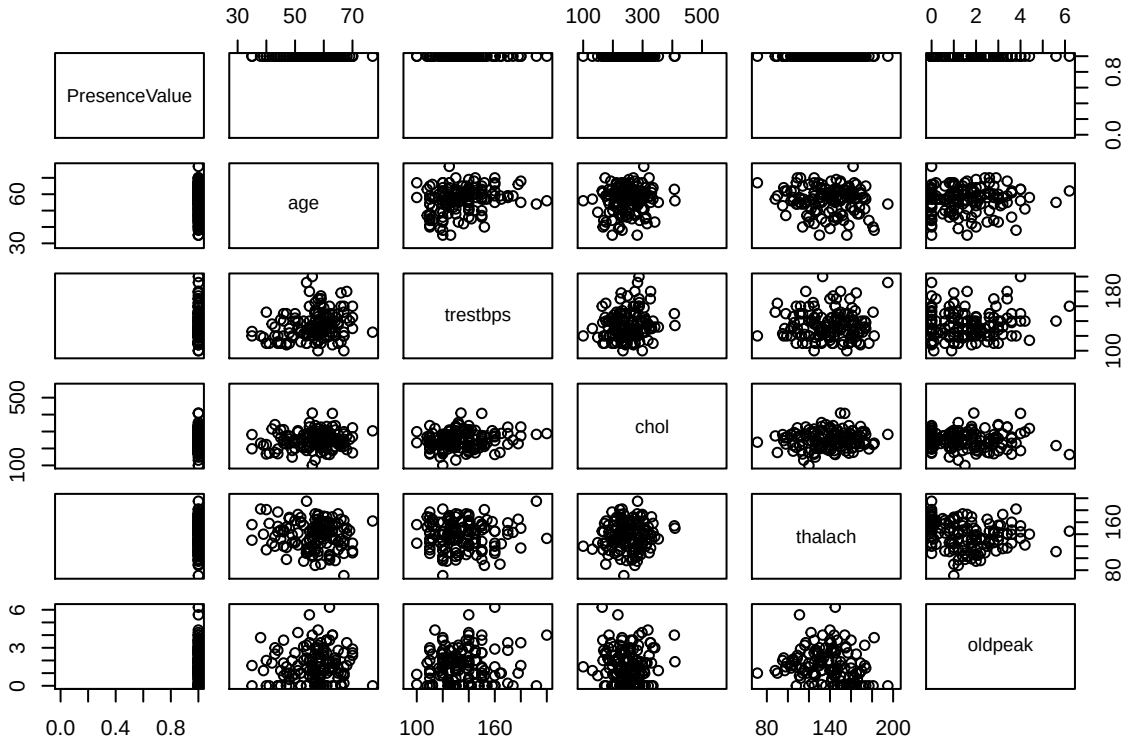

Presence of Heart Disease by RestingECG
Count
75
50
25
0
left ventricular hypertrophy
normal
ST−T wave abnormality
RestingECG
Presence
healthy
presence



Method - Data exploration - Continuous Variables

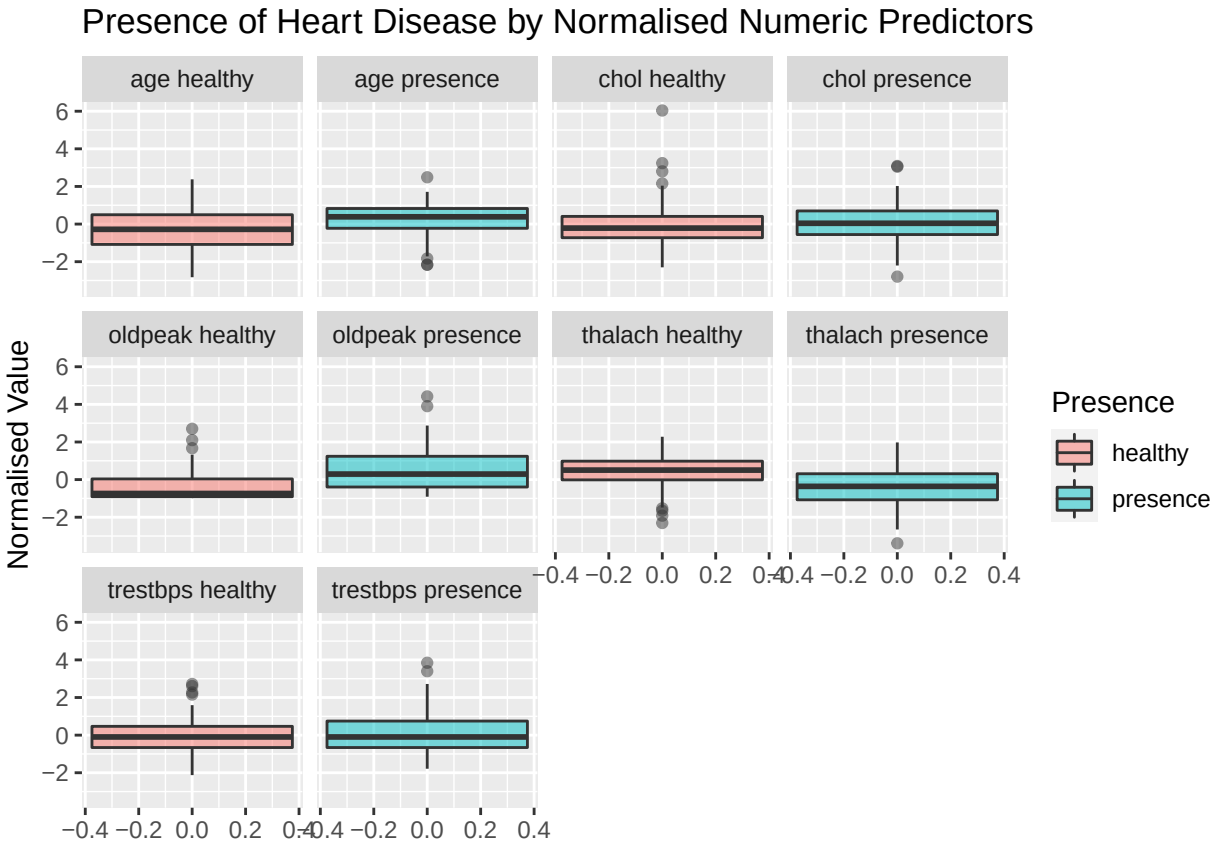
For continuous variables lets look and see whether any of these are highly correlated to each other. We will examine this further with PCA later when looking at their multicollinearity and "components".

Pairs visualisation for correlation There doesnt seem to be much correlation between our continuous variables.



Correlation values This is confirmed when looking at the correlation values below. This would suggest that PCA/PCR/PLS analysis would not be very helpful.

Normalised Continous Variables In order to determine which of the continuous variables has the best predictive power we can look at them as boxplots and see which of these is significantly different for a positive/negative diagnosis of heart disease.



Predictors chosen for models

For models where having a lower number of predictors improves its reliability and reduces the chance of these filling the "noise" we will reduce these by using those selected below.

Factor variables Based on the graphics shown above we have seleted the below variables as being potentially statistically different. Though this is not limited to just the below we have made a observational choice.

1. (ca) - number of major vessels (0-3) colored by flourosopy
2. (cp) - chest pain type
3. (thal) - Form of thalassemia

Continuous variables From the continuous predictors there is the greatest difference seen within "age", "oldpeak" and "thalach".

4. (age) - age in years
5. (oldpeak) - ST depression induced by exercise relative to rest
6. (thalach) - maximum heart rate achieved

Results - Building a model to predict heart disease

Model 1 - K Means

K-means works by distance between variables, as such categorical variables will not work within the model so we shall remove these first.

We will be using the continuous variables namely; age,trestbps, chol, thalach, oldpeak along with our reassigned outcome PresenceValue which now only has two results being 0 (absence) and 1 (presence) of heart disease.

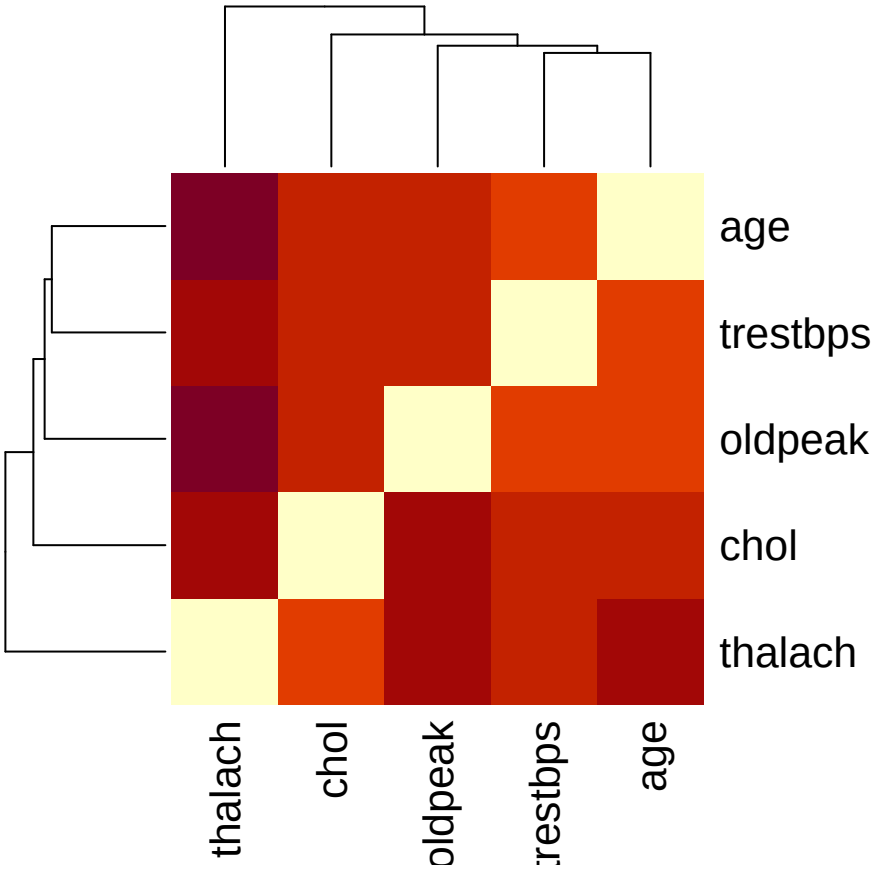
In order to perform cluster analysis on the data we must also standardise the data to make variables comparable.

First lets calculate the average distance between the predictors of our outcome.

```
## [1] 2.823127
```

Heatmap of relationships

Now lets create a heatmap of the relationship between features using the scaled matrix.



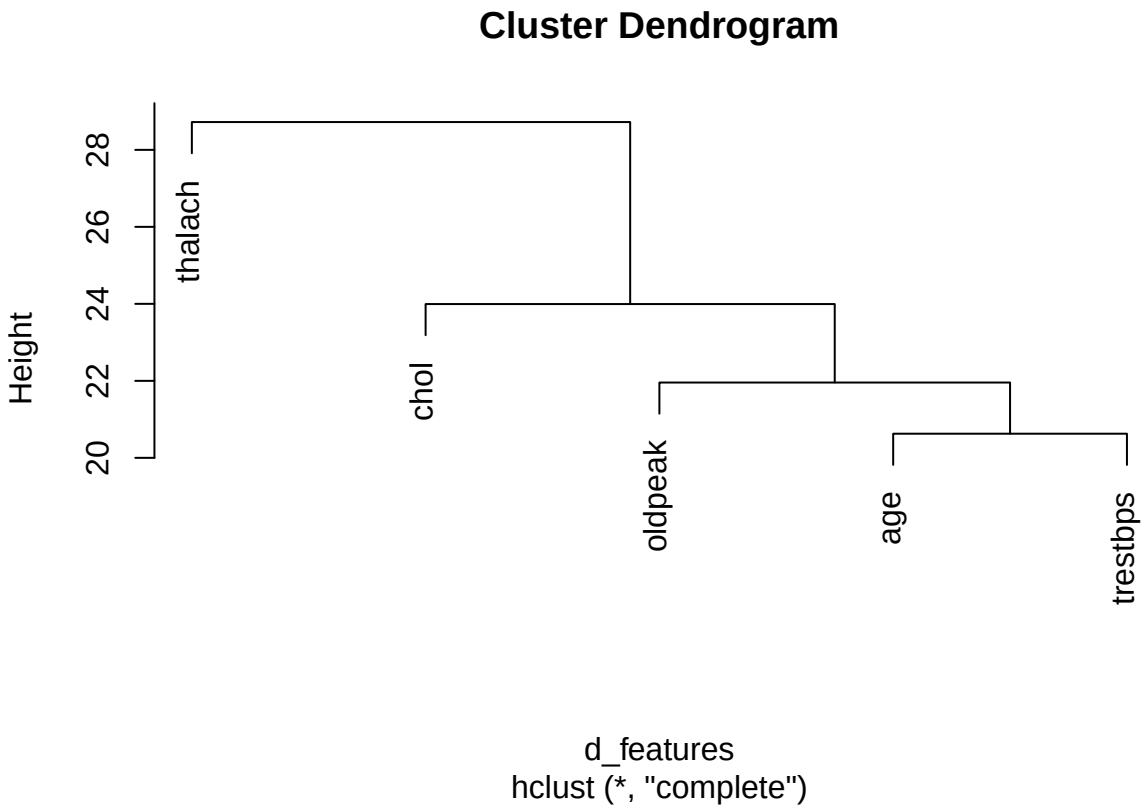
We can see the highest distance between thalach and age. It makes sense that heart rate would relate to age and ST depression induced by exercise (when also considering oldpeak).

Hierarchical Clustering of Features

We want an algorithm to define groups from our predictors. Hierarchical clustering will define each observation as a separate group, then the two closest groups are joined into a group iteratively until there is just one group including all the observations.

The hclust function implements this algorithm and it takes a distance as input.

We can see from the visualisation that thalach is furthest from age and tresbps.



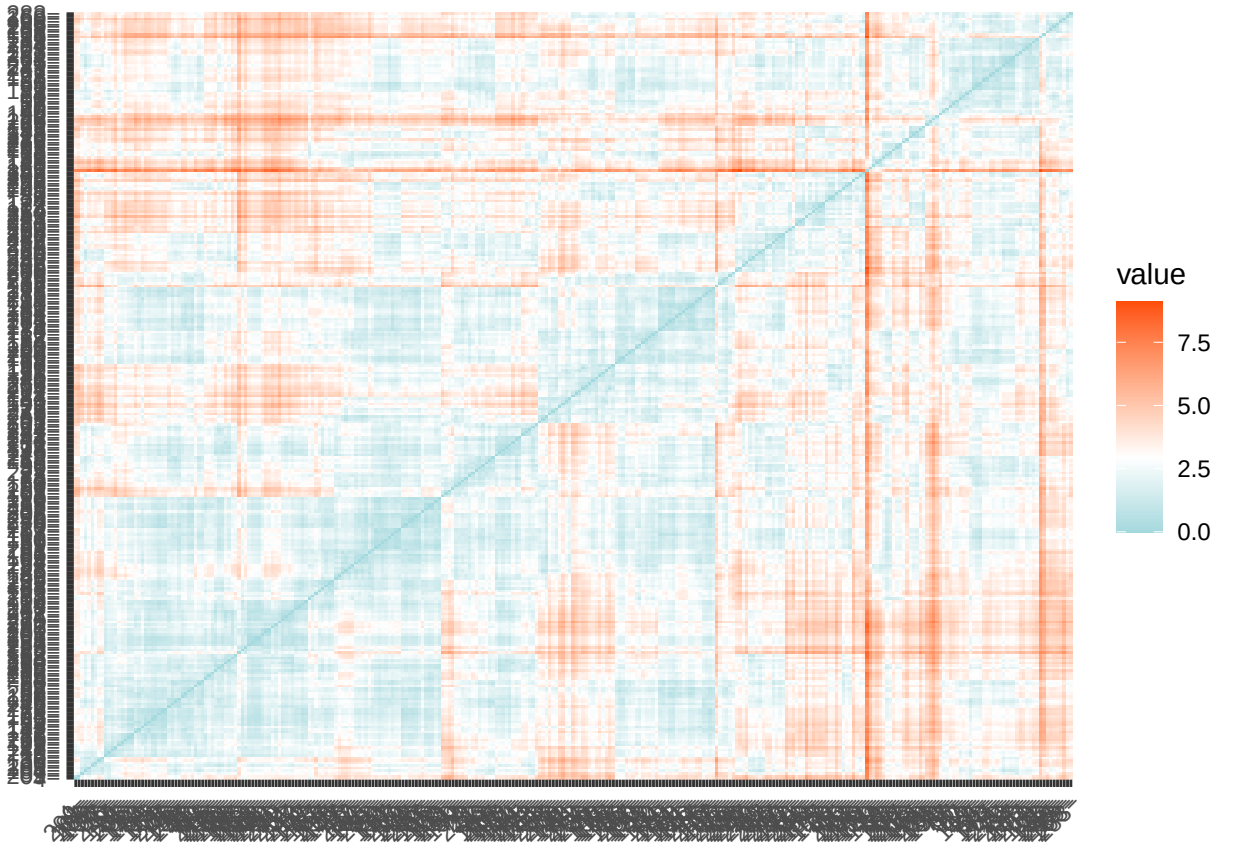
Another way to view this would be in groupings. We do not have many features here to pick from so shall only assign k = 3.

```
## $`1`
## [1] "age"      "trestbps" "oldpeak"
##
## $`2`
## [1] "chol"
##
## $`3`
## [1] "thalach"
```

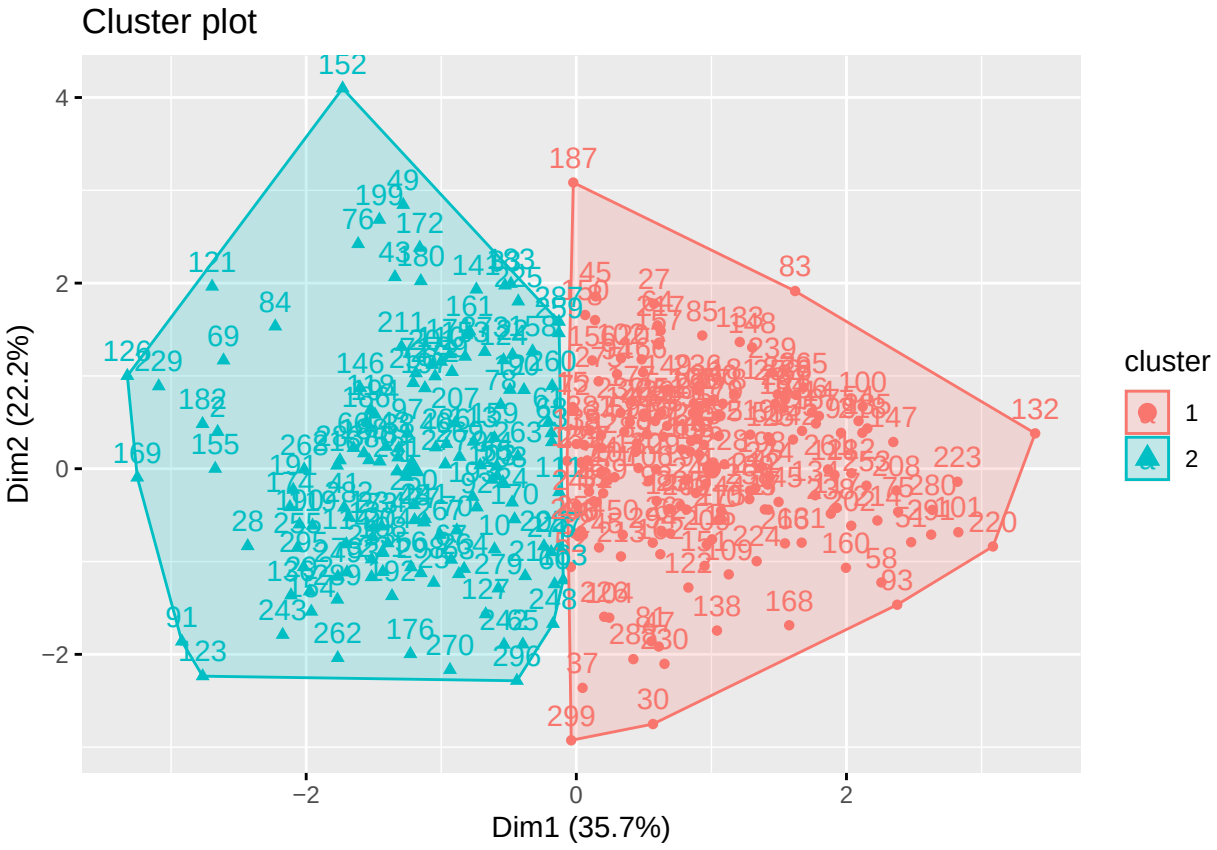
K Means Clustering - Visualisation of K

The Factoextra package provides us with some neat visualisations to use within R.

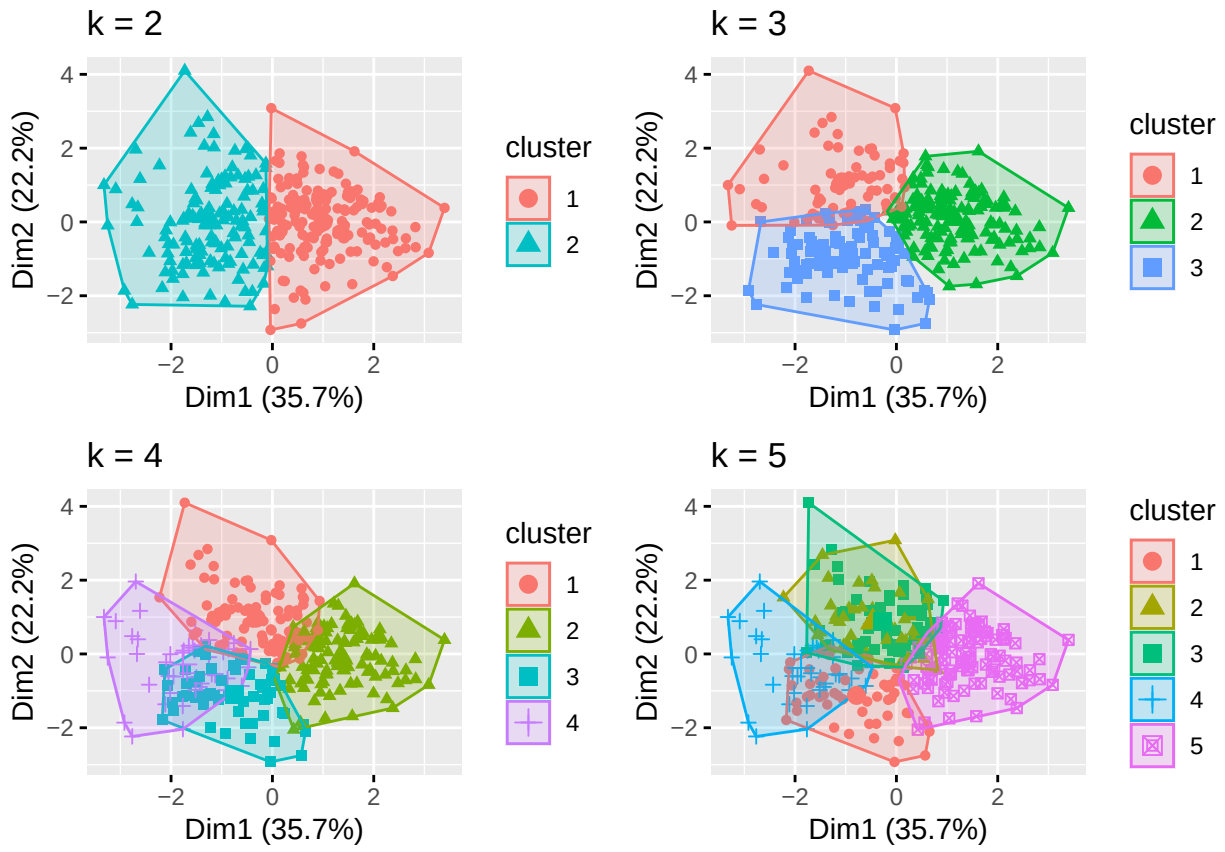
For example we can illustrate which observations have large dissimilarities (red) versus those that appear to be fairly similar (teal). Please note that this looks at all observations as such is only useful for a quick assessment on how similar/dissimilar the data is overall.



We can also calculate k-means within this package and visualise the clustering within the predictors. Let's start by determining 2 clusters and run the configuration 25 times (the best one being reported on).



Let's run a few more of these but with higher center values and compare 2, 3, 4, and 5 centers.

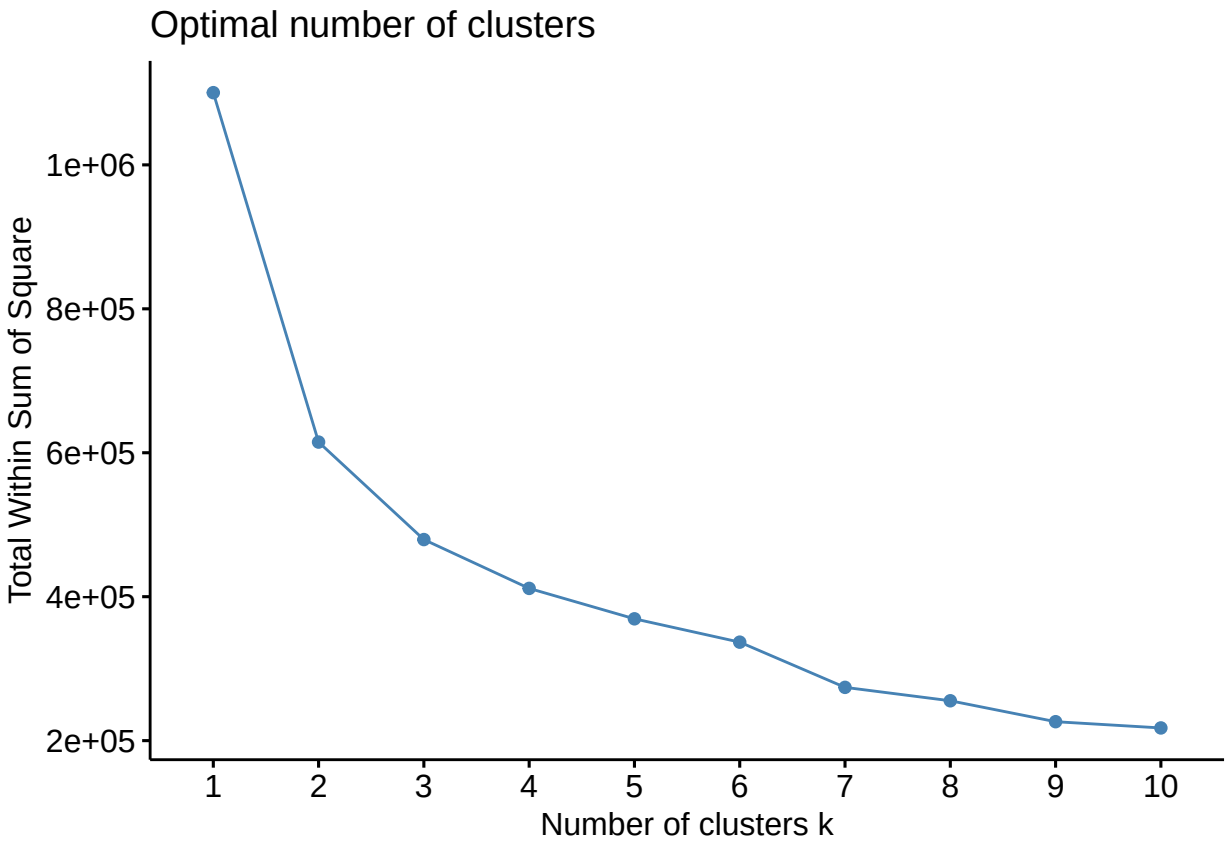


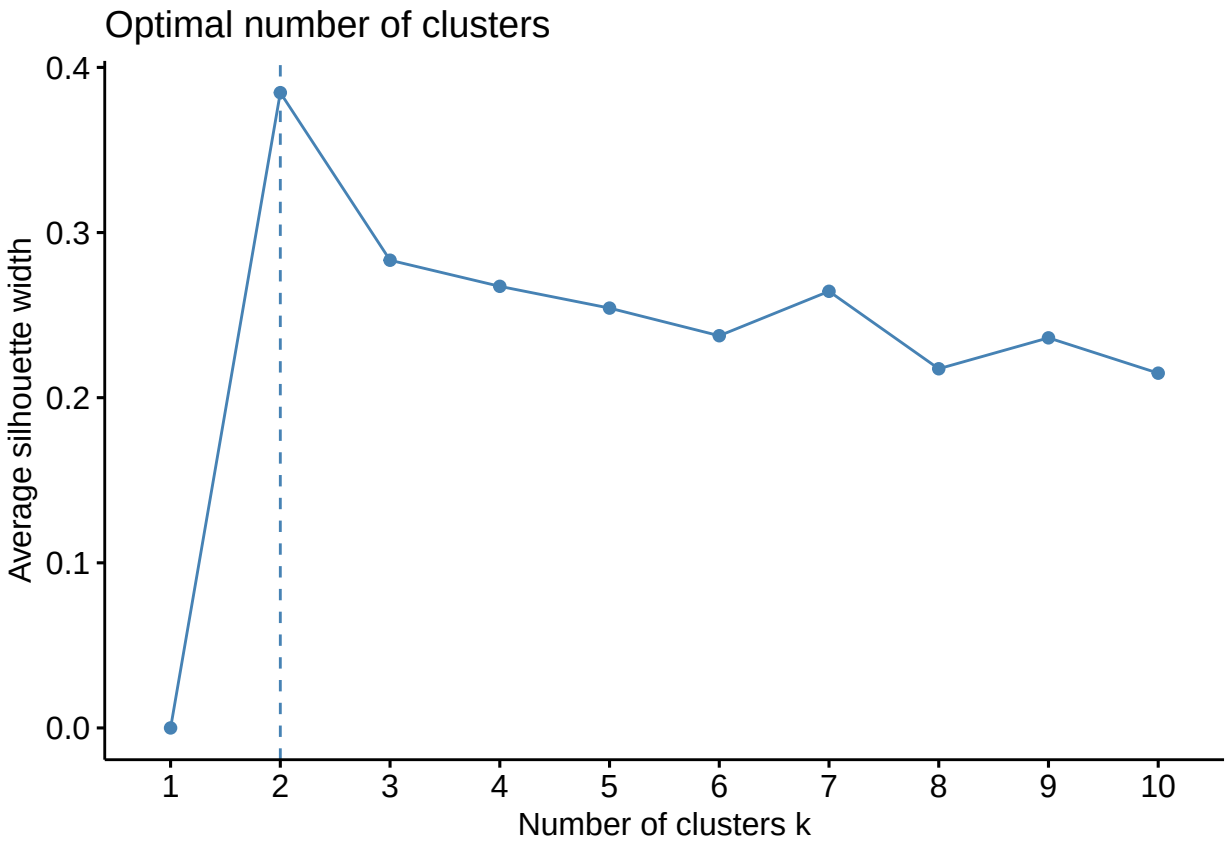
However, although we have a visual representation of where the dilineations occur this does not actually tell us the optimal number of clusters (though admittedly it is likely two).

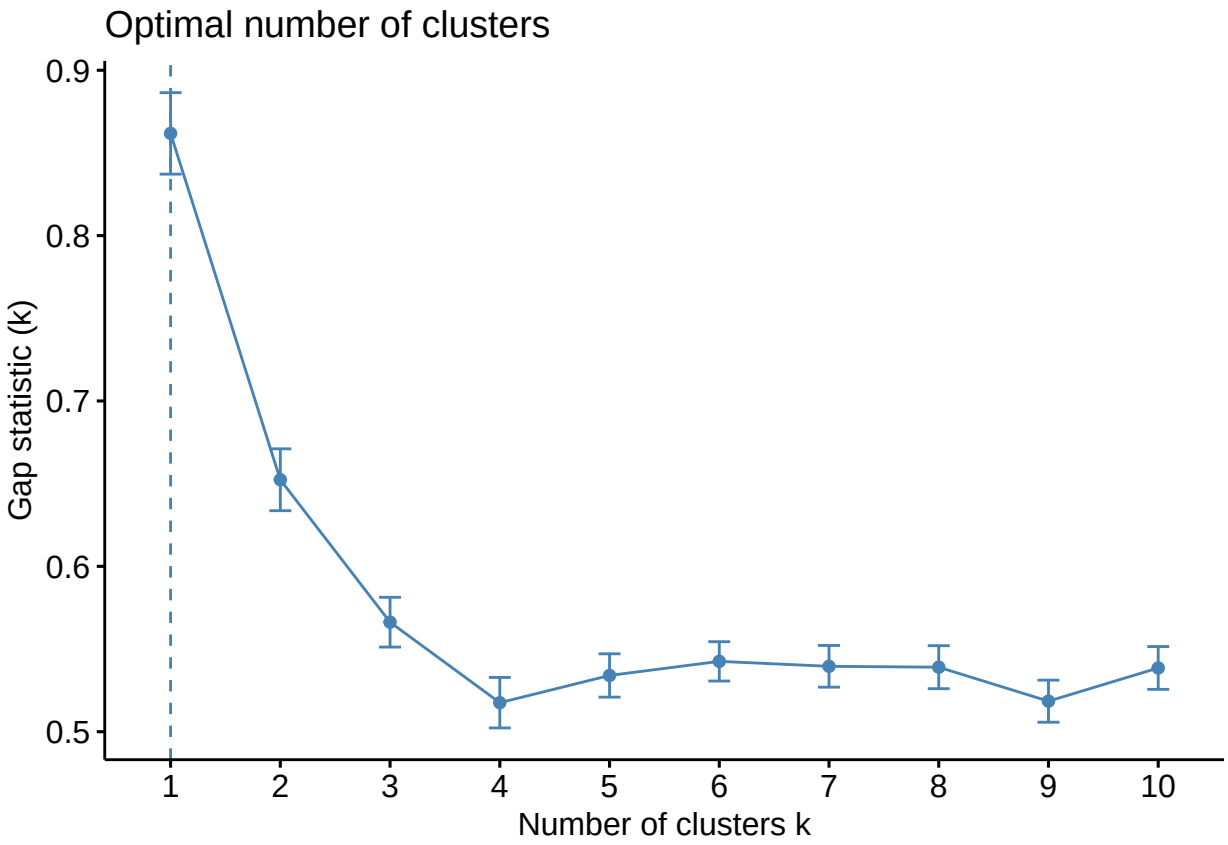
We can instead run three methods to confirm the ideal number of clusters.

1. Elbox method
2. Silhoutte method
3. Gap statistic method

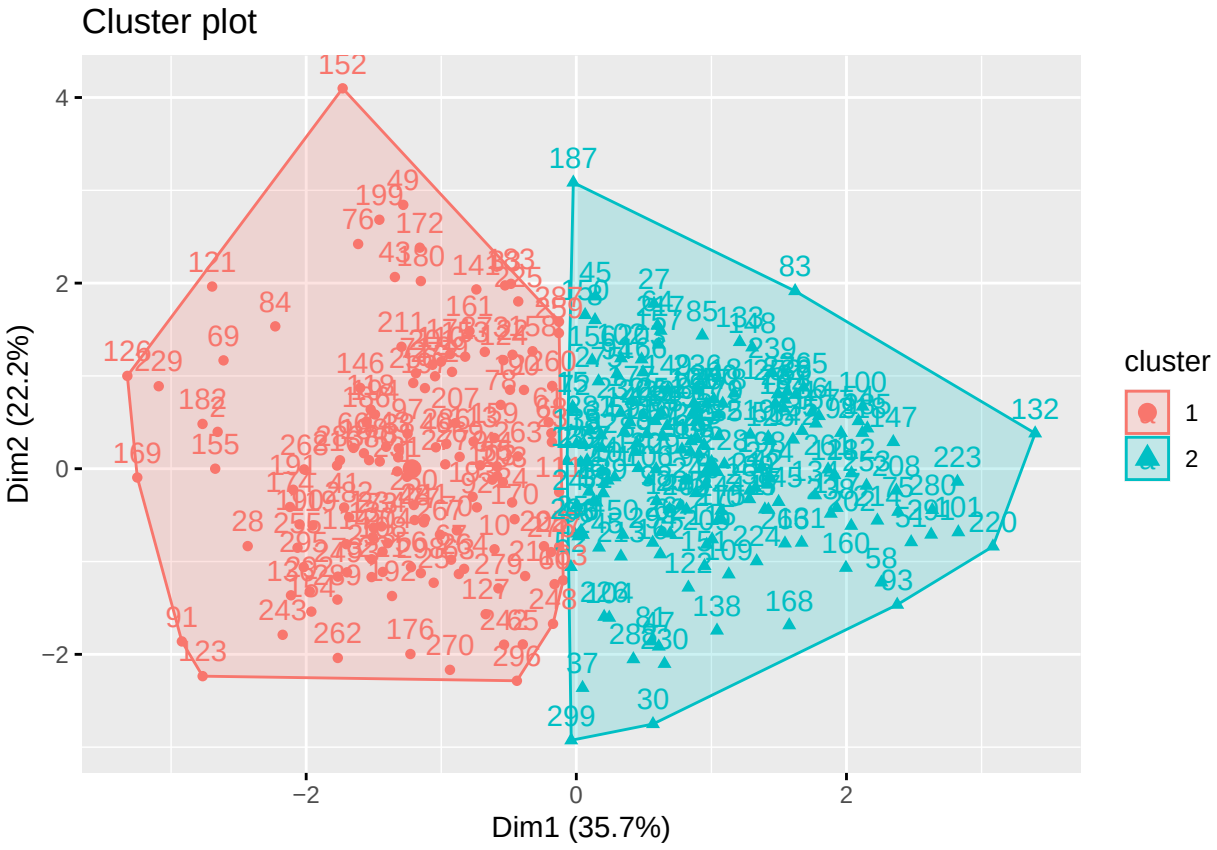
All three methods below show us that two is the optimal number (for elbox this is where the "bend" is).


Optimal number of clusters
Total Within Sum of Square
1e+06
8e+05
6e+05
4e+05
2e+05
1
2
3
4
5
6
7
8
9
10
Number of clusters k


Optimal number of clusters
Average silhouette width
0.4
0.3
0.2
0.1
0.0
1
2
3
4
5
6
7
8
9
10
Number of clusters k



So we can return to our first graph. The optimal number of clusters for our predictors is two.




```
## K-means clustering with 2 clusters of sizes 135, 164
##
## Cluster means:
##          age   trestbps       chol    thalach    oldpeak
## 1  0.6796612  0.4513364  0.2963799 -0.6440328  0.5599573
## 2 -0.5594772 -0.3715269 -0.2439713  0.5301490 -0.4609404
##
## Clustering vector:
##   [1] 1 1 1 2 2 2 1 2 1 1 2 1 2 2 2 2 2 2 2 2 1 1 1 1 1 2 2 1 2 2 1 2 1 2 2 2 2
##  [38] 1 1 1 1 2 1 1 2 2 2 1 1 2 2 2 2 2 2 1 1 2 2 2 1 1 2 1 2 1 1 1 1 1 1 1 2 1 2
##  [75] 2 1 1 1 2 1 2 2 2 1 2 2 2 2 2 2 1 1 1 2 2 2 2 1 2 2 2 2 2 1 2 2 2 2 1 2 1 1
## [112] 2 1 1 2 2 2 1 1 2 1 2 1 1 2 1 1 2 2 2 2 2 2 2 2 1 1 2 2 2 1 2 1 2 2 1 2 2
## [149] 2 2 2 1 1 1 1 2 2 1 1 2 1 2 1 2 2 2 2 2 1 1 1 1 1 1 2 1 2 2 2 1 2 1 1 2 2
## [186] 1 2 1 2 1 1 1 1 1 2 2 1 2 1 2 1 2 2 1 1 2 1 2 2 2 1 2 2 2 2 1 2 2 2 2 1 2
## [223] 2 2 1 2 1 2 1 2 1 2 1 1 2 2 2 2 2 2 1 1 1 2 2 2 2 1 1 2 2 2 1 1 2 2 1 2 1
## [260] 1 2 1 1 1 2 2 1 1 1 1 2 1 1 2 2 1 1 2 1 2 2 1 1 2 2 1 1 2 1 2 2 1 1 2 1 1
## [297] 2 1 2
##
## Within cluster sum of squares by cluster:
## [1] 631.2421 494.0415
##  (between_SS / total_SS =  24.5 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

From the above we can see our final model using K=2 explains 24.5% of the variance in our data set. This means that although we are about to perform k-means as a method of prediction, the predictive ability is actually fairly poor.

K Means Clustering - Modelling the Algorithym

Define a k-means predict function Let's create a k-means predict function. The predict_kmeans() function defined here takes two arguments - a matrix of observations x and a k-means object k - and assigns each row of x to a cluster from k.

```
predict_kmeans <- function(x, k) {
  centers <- k$centers    # extract cluster centers
  # calculate distance to cluster centers
  distances <- sapply(1:nrow(x), function(i){
    apply(centers, 1, function(y) dist(rbind(x[i,], y)))
  })
  max.col(-t(distances))  # select cluster with min distance to center
}
```

For this model we are using the predictors; age, trestbps, chol, thalach and oldpeak.

K Means Clustering - Testing and training

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0  8 16
##          1 21  9
##
##                Accuracy : 0.3148
##                  95% CI : (0.1952, 0.4555)
##     No Information Rate : 0.537
##     P-Value [Acc > NIR] : 0.9997
##
##                   Kappa : -0.3592
##
##  Mcnemar's Test P-Value : 0.5108
##
##             Sensitivity : 0.2759
##             Specificity : 0.3600
##          Pos Pred Value : 0.3333
##          Neg Pred Value : 0.3000
##              Prevalence : 0.5370
##          Detection Rate : 0.1481
##    Detection Prevalence : 0.4444
##       Balanced Accuracy : 0.3179
##
##        'Positive' Class : 0
##
```

This means that k-means is giving us an accuracy of 0.315, correctly identifies 0.276 of patients without heart disease (Sensitivity) and 0.36 of patients with heart disease (Specificity).

Previously we stated that only 24.5% of the variance is explained by the numeric variables selected. We can demonstate this variability by running the test again but with seed = 3. Here we now have 68.5% accuracy due to chance. We shall therefore keep the first result which is more reflective of the models predictive power.

```
## [1] 0.6851852
```

Now we can add our results to a table to be presented at the end of our experiment. We have rounded these to 3dp.

Method	Accuracy	LowerCI	UpperCI	Sensitivity	Specificity	RMSE	Rsquare
Model 1 - K Means	0.315	0.195	0.456	0.276	0.36	0.561	0.134

Model 2-8 - Setting K-fold cross validation control

In the next 7 models we will utilise k-fold cross validation which we will set to run with 10% sampling (running 10 times) and repeat the whole process 3 times. This will increase the accuracy of our final model for each method.

```
# Repeated K-fold cross-validation
control <- trainControl(method = "repeatedcv",
                        number = 10,
                        repeats = 3)
```

Model 2 - Partial Least Squares ("PLS")

Before we jump into PLS we will firstly look at PCA analysis to show why this would not be a good method of prediction for our data set.

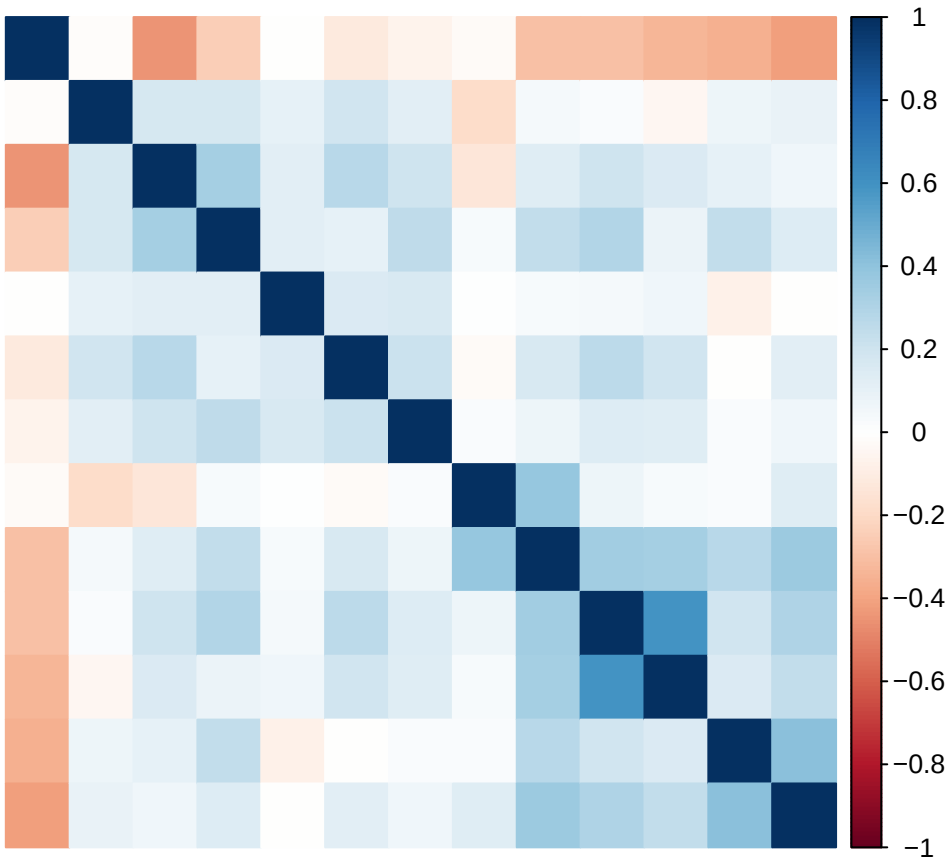
Firstly we will reshape the data for both PLS and PCA as this will need to be in the same format. We will revert factors back into numeric values.

Visualisation of Principal Component Analysis ("PCA")

We will now use Principal Component Analysis ("PCA") to identify components within our variables to try to explain the variation.

In PCA the data should first be standardised as they have different units and variances. If we do not do this then the first component will be mainly determined by variables with the largest variance. We will do this below via the "center" and "scale" arguments.

We can see below that there is some correlation between the variables. Though I would say this is not extreme and therefore doesn't bode well for PCA having statistical relevance to this analysis as it relies on creating components from correlations of variables.



In reality PCA/PCR is more useful for data sets with larger number of predictors and when these have large correlation.

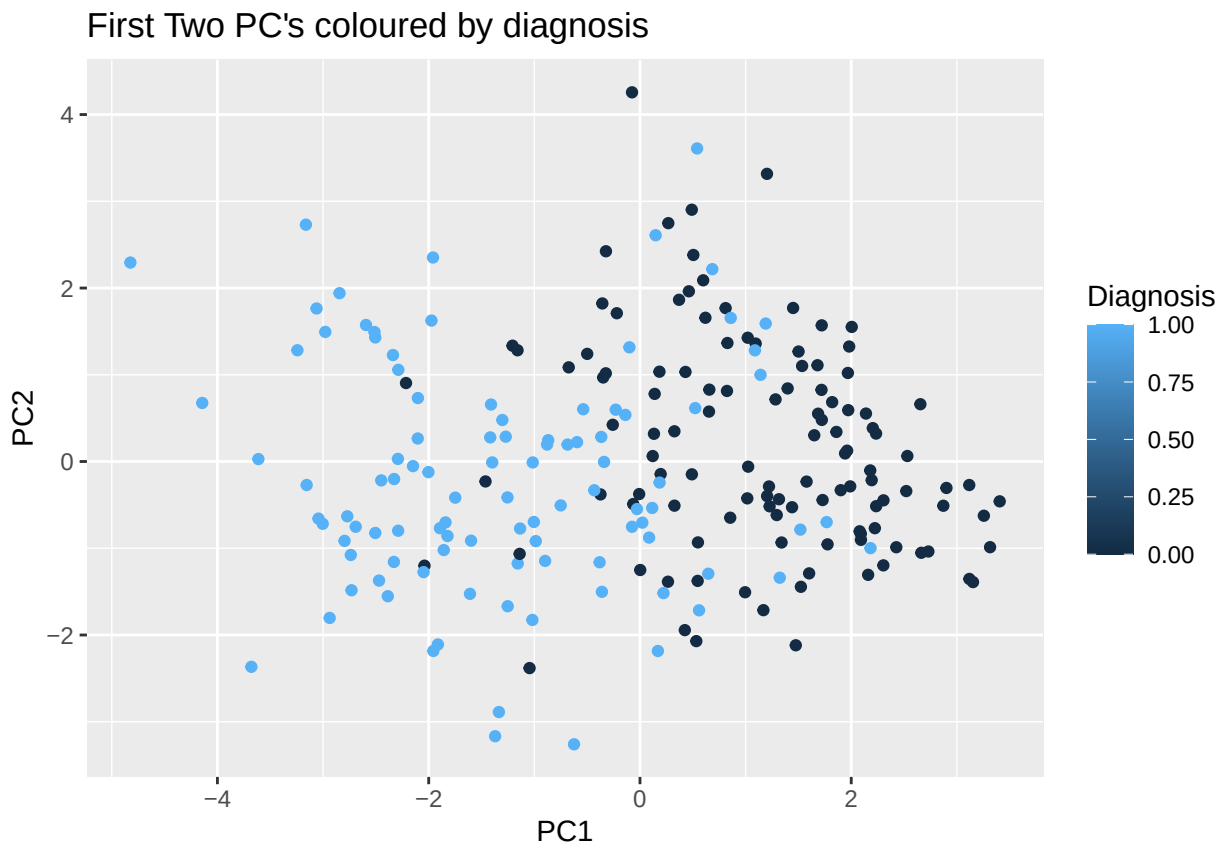
This method intends to reduce the dimensions of our predictors, uses components to try to avoid the multi-collinearity between the predictors and reduce the chance of an overfitted model.

We set center and scale to true in order to normalise the data. This will center the data around its mean (reducing each columns mean to zero) and therefore scaling all predictors around each other so they don't have more weight than each other.

Running PCA we can see that the cumulative proportion is at ~47% with three principal components ("PC's"), ideally we would look for a higher figure as again using too many PC's can lead to over fitting.

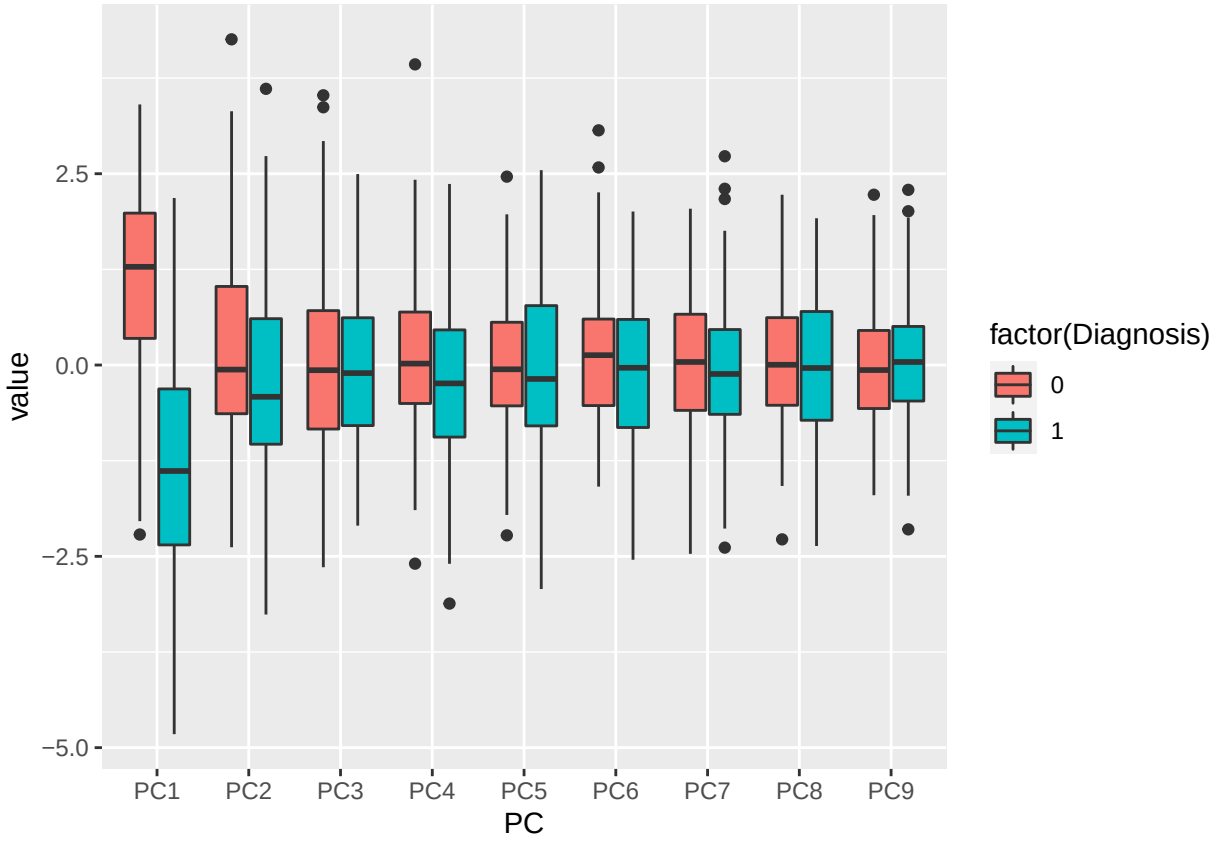
```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     1.7782 1.2896 1.13314 1.06612 0.98343 0.92421 0.91727
## Proportion of Variance 0.2432 0.1279 0.09877 0.08743 0.07439 0.06571 0.06472
## Cumulative Proportion  0.2432 0.3712 0.46993 0.55736 0.63176 0.69746 0.76218
##                           PC8     PC9    PC10    PC11    PC12    PC13
## Standard deviation     0.87763 0.79087 0.73159 0.67392 0.62693 0.55989
## Proportion of Variance 0.05925 0.04811 0.04117 0.03494 0.03023 0.02411
## Cumulative Proportion  0.82143 0.86955 0.91072 0.94565 0.97589 1.00000
```

Let's plot the first two principal components to get a visual representation of this. We will colour the outcome (absence = 0/heart disease = 1).

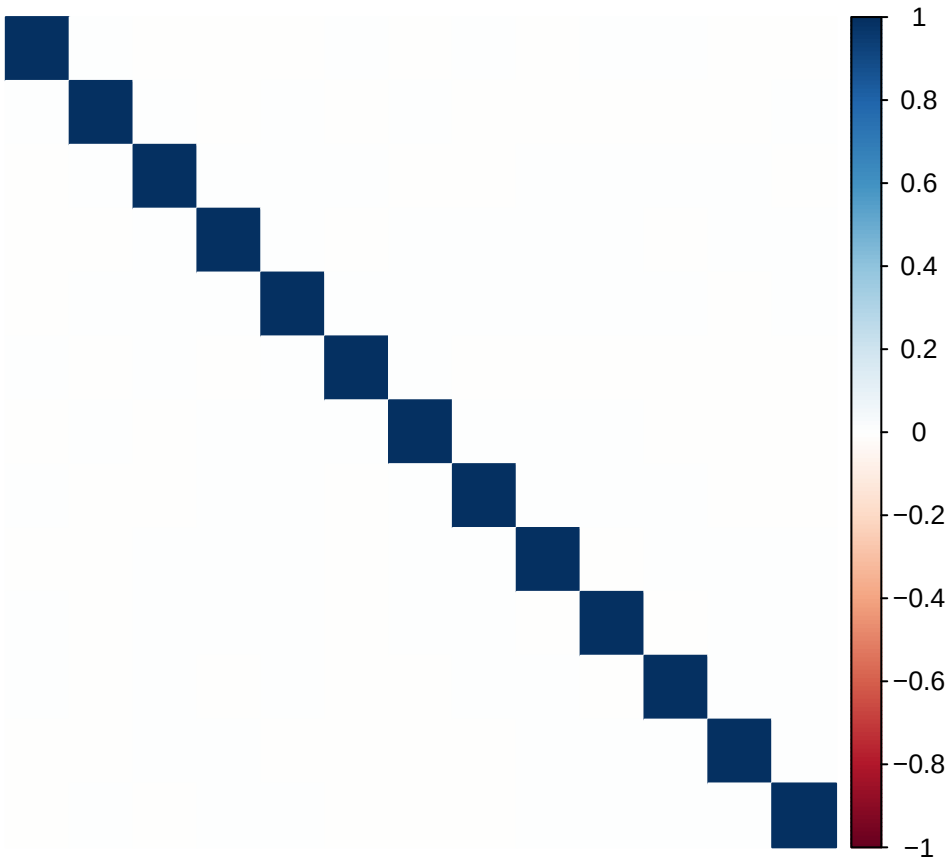


There is high overlap of the two level diagnosis, showing the first two components alone would not be very useful to determine the presence of heart disease.

We can also create boxplots of each of the first nine PC's to show which are statistically different within the diagnosis. This shows that the first PC is quite useful in its predictive power but following from that the rest have little value.

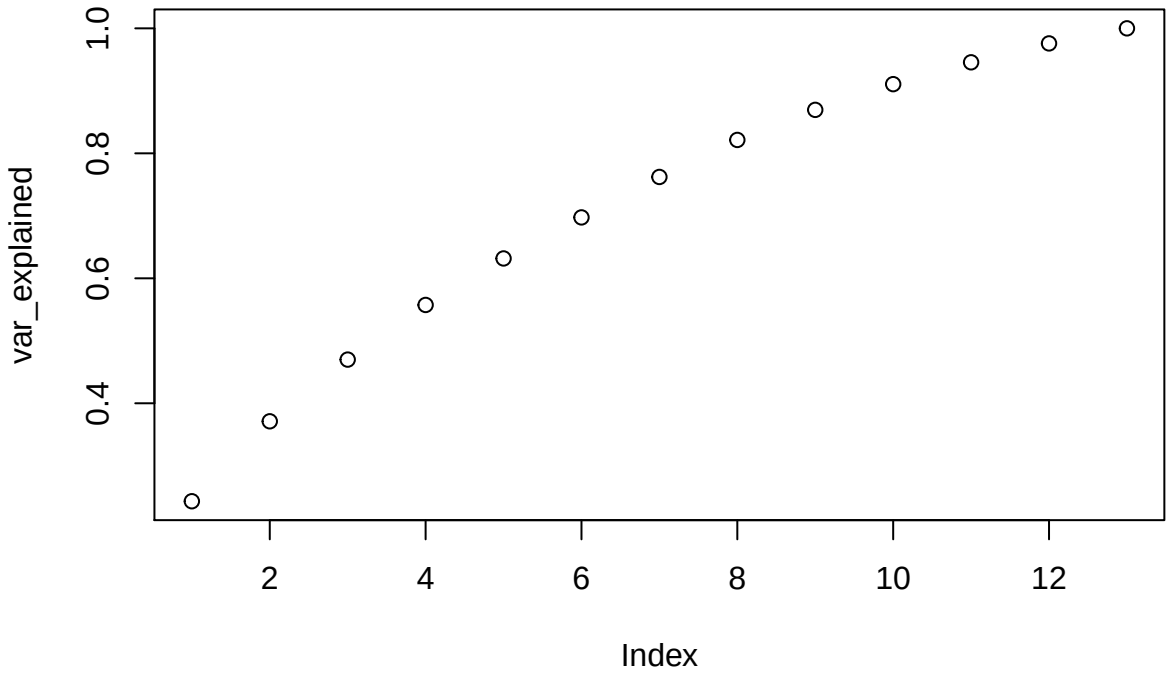


Since the principal components are orthogonal, there is no correlation whatsoever. The correlation plot is perfectly white, apart from autocorrelation.



The below plot below shows what percent of variance has been explained for each number of principal components (aggregate variance explained). Ideally we would be seeing a steep curve, unfortunately the smoothness shows a lack of variance explained by the first few components.

Overall PCA confirms the lack of multicollinearity between our predictors.



Model 2 - Partial Least Squares ("PLS")

Partial least squares is a method, similar to principal component regression, that chooses the components to include based on the correlation with the outcome . This makes the method non-linear and thus computationally less attractive.

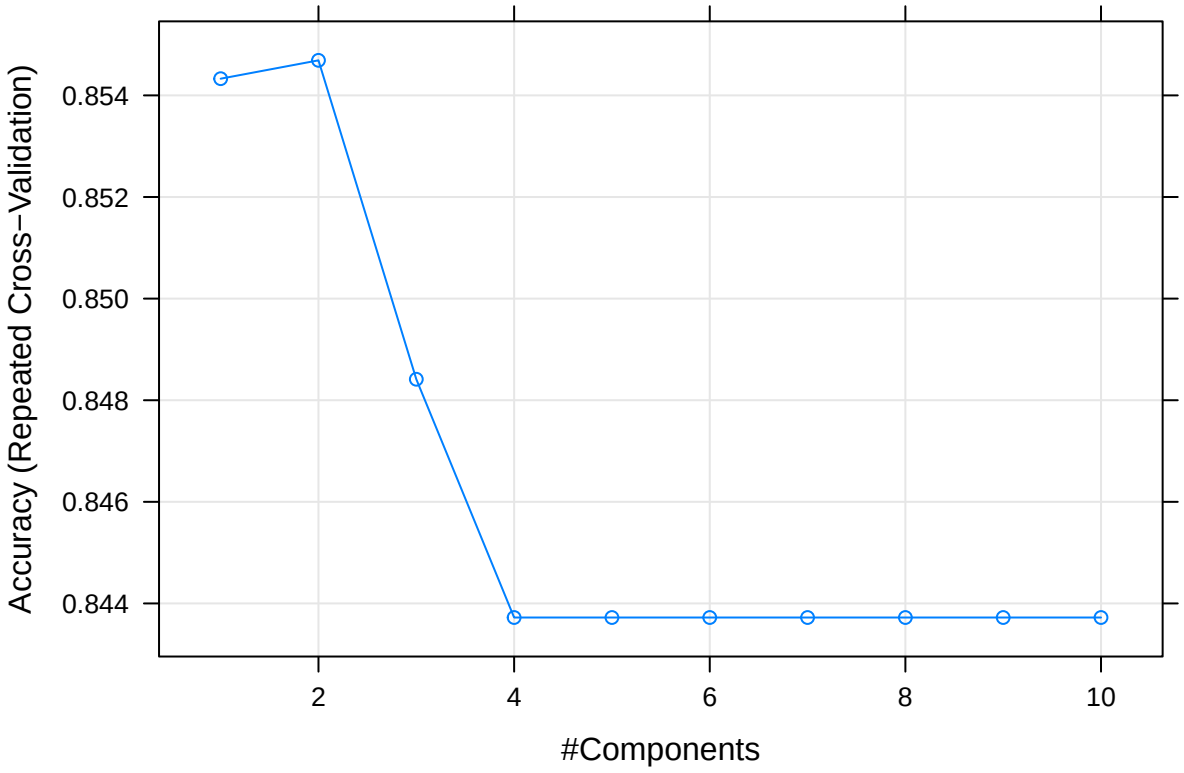
Since our PCA analysis didn't seem too successful it is therefore unlikely to be our best method.

However, we shall still attempt it, if only for comparison. So let's build our PLS model.

```
# Build the model on training set
set.seed(1)
model <- train( PresenceValue ~ . , data = pc.train_set , method = "pls",
                scale = TRUE,
                trControl = control,
                metric = "Accuracy",
                tuneLength = 10  )
```

Our plot below shows that 2 components gives us the highest accuracy (also confirmed by best tune).

```
# Plot model RMSE vs different values of components
plot(model)
```




```
# Print the best tuning parameter ncomp that
# minimize the cross-validation error, RMSE
model$bestTune
```

```
##   ncomp
## 2     2
```

```
# Make predictions
pls_preds <- predict(model, newdata = pc.test_set)

# Change preds into numeric not factor for analysis
pls_preds <- as.numeric(as.character(pls_preds))


Accuracy <- mean(pls_preds == pc.test_set_y)

confusionmatrix.results <- confusionMatrix(factor(kmeans_preds), factor(pc.test_set_y))

confusionmatrix.results # This also contains the accuracy that we calculated above.
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 21  9
```

```
##          1  8 16
##
##                Accuracy : 0.6852
##                  95% CI : (0.5445, 0.8048)
##     No Information Rate : 0.537
##     P-Value [Acc > NIR] : 0.01929
##
##                   Kappa : 0.3651
##
##  Mcnemar's Test P-Value : 1.00000
##
##             Sensitivity : 0.7241
##             Specificity : 0.6400
##          Pos Pred Value : 0.7000
##          Neg Pred Value : 0.6667
##              Prevalence : 0.5370
##          Detection Rate : 0.3889
##    Detection Prevalence : 0.5556
##       Balanced Accuracy : 0.6821
##
##        'Positive' Class : 0
##
```

```
# Accuracy <- confusionmatrix.results$overall[1] Alternative way of showing accuracy already calculated
LowerCI <- confusionmatrix.results$overall[3]
UpperCI <- confusionmatrix.results$overall[4]

Sensitivity <- confusionmatrix.results$byClass[1]
Specificity <- confusionmatrix.results$byClass[2]

# Here we add the results to a table.
Accuracy_Results <- rbind(Accuracy_Results,
                          tibble(
                            Method = "Model 2 - PLS",
                             Accuracy = round(Accuracy, 3),
                            LowerCI = round(LowerCI, 3),
                            UpperCI = round(UpperCI,3),
                            Sensitivity = round(Sensitivity,3),
                            Specificity = round(Specificity,3),
                            RMSE = round(caret::RMSE(pls_preds, pc.test_set_y ),3),
                            Rsquare = round(caret::R2(pls_preds, pc.test_set_y ),3)              ))

Accuracy_Results %>% knitr::kable()
```

Method	Accuracy	LowerCI	UpperCI	Sensitivity	Specificity	RMSE	Rsquare
Model 1 - K Means	0.315	0.195	0.456	0.276	0.36	0.561	0.134
Model 2 - PLS	0.815	0.544	0.805	0.724	0.64	0.430	0.394

We can see that the specifity (detecting the presence of heart disease) is quite low at 64%. We expected it wouldn't be high though this is now confirmed.

We had previously identified thalach , ca, oldpeak, age, cp and thal predictors as potentially useful in

determining accurate prediction models. We shall use this in all remaining models.

Model 3 - General Logistic Regression ("GLM")

Here we will apply logistic regression to our data which will assign numeric values of 0 and 1 to the outcomes.

```
set.seed(1)
train_glm <- train(PresenceValue ~  thalach + ca + oldpeak + age + cp + thal  ,
                   data = train_set,
                   trControl = control,
                   method = "glm" ,
                   family=binomial) # The dependant variable we are trying to pick is binomial (has two
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 24  5
##          1  5 20
##
##                Accuracy : 0.8148
##                  95% CI : (0.6857, 0.9075)
##     No Information Rate : 0.537
##     P-Value [Acc > NIR] : 1.899e-05
##
##                   Kappa : 0.6276
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8276
##             Specificity : 0.8000
##          Pos Pred Value : 0.8276
##          Neg Pred Value : 0.8000
##              Prevalence : 0.5370
##          Detection Rate : 0.4444
##    Detection Prevalence : 0.5370
##       Balanced Accuracy : 0.8138
##
##        'Positive' Class : 0
##
```

Method	Accuracy	LowerCI	UpperCI	Sensitivity	Specificity	RMSE	Rsquare
Model 1 - K Means	0.315	0.195	0.456	0.276	0.36	0.561	0.134
Model 2 - PLS	0.815	0.544	0.805	0.724	0.64	0.430	0.394
Model 3 - GLM	0.815	0.686	0.907	0.828	0.80	0.430	0.394

GLM has provided good results, with the highest confidence intervals seen thus far and both sensitivity and specificity are at/above 80%. This already beats the historical attempts.

Model 4 - Linear Discriminant Analysis ("LDA")

Linear Discriminant Analysis ("LDA") is a specific case of the general generative model, Naive Bayes. Bayes theorem tells us that knowing the distribution of the predictors X may be useful.

LDA estimates the probability that a new set of inputs belongs to every class. The output class is the one that has the highest probability.

```
set.seed(1)
train_lda <- train(PresenceValue ~  thalach + ca + oldpeak + age + cp + thal ,
                   data = train_set,
                   trControl = control,
                   method = "lda")

lda_preds <- predict(train_lda, test_set)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 24  6
##          1  5 19
##
##                Accuracy : 0.7963
##                  95% CI : (0.6647, 0.8937)
##     No Information Rate : 0.537
##     P-Value [Acc > NIR] : 6.822e-05
##
##                   Kappa : 0.5892
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8276
##             Specificity : 0.7600
##          Pos Pred Value : 0.8000
##          Neg Pred Value : 0.7917
##              Prevalence : 0.5370
##          Detection Rate : 0.4444
##    Detection Prevalence : 0.5556
##       Balanced Accuracy : 0.7938
##
##        'Positive' Class : 0
##
```

Method	Accuracy	LowerCI	UpperCI	Sensitivity	Specificity	RMSE	Rsquare
Model 1 - K Means	0.315	0.195	0.456	0.276	0.36	0.561	0.134
Model 2 - PLS	0.815	0.544	0.805	0.724	0.64	0.430	0.394
Model 3 - GLM	0.815	0.686	0.907	0.828	0.80	0.430	0.394
Model 4 - LDA	0.796	0.665	0.894	0.828	0.76	0.451	0.348

We should expect to see fairly similar results with GLM, LDA and QDA. LDA's accuracy is very high though its specificity for detecting the disease is a little lower.

Model 5 - Quadratic Discriminant Analysis ("QDA")

QDA is slightly different to LDA, where in LDA a linear boundary is needed between classifiers, QDA is used to find non-linear boundaries.

```
set.seed(1)
train_qda <- train(PresenceValue ~ thalach + ca + oldpeak + age + cp + thal  ,
                   data = train_set,
                   method = "qda",
                   na.rm=TRUE )

qda_preds <- predict(train_qda, test_set)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 22  5
##          1  7 20
##
##                Accuracy : 0.7778
##                  95% CI : (0.644, 0.8796)
##     No Information Rate : 0.537
##     P-Value [Acc > NIR] : 0.0002203
##
##                   Kappa : 0.5556
##
##  Mcnemar's Test P-Value : 0.7728300
##
##             Sensitivity : 0.7586
##             Specificity : 0.8000
##          Pos Pred Value : 0.8148
##          Neg Pred Value : 0.7407
##              Prevalence : 0.5370
##          Detection Rate : 0.4074
##    Detection Prevalence : 0.5000
##       Balanced Accuracy : 0.7793
##
##        'Positive' Class : 0
##
```

Method	Accuracy	LowerCI	UpperCI	Sensitivity	Specificity	RMSE	Rsquare
Model 1 - K Means	0.315	0.195	0.456	0.276	0.36	0.561	0.134
Model 2 - PLS	0.815	0.544	0.805	0.724	0.64	0.430	0.394
Model 3 - GLM	0.815	0.686	0.907	0.828	0.80	0.430	0.394
Model 4 - LDA	0.796	0.665	0.894	0.828	0.76	0.451	0.348
Model 5 - QDA	0.778	0.644	0.880	0.759	0.80	0.471	0.310

Model 6 - Local Weighted Regression ("Loess")

We can use Loess, this method is non-linear and tries to down weight large residuals and refit the data. In essence this reduces the skew by our outliers.

```
set.seed(1)
train_loess <- train(PresenceValue ~  thalach + ca + oldpeak + age + cp + thal ,
                     data = train_set,
                     trControl = control,
                     method = "gamLoess")

loess_preds <- predict(train_loess, test_set)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 25  5
##          1  4 20
##
##                Accuracy : 0.8333
##                  95% CI : (0.7071, 0.9208)
##     No Information Rate : 0.537
##     P-Value [Acc > NIR] : 4.712e-06
##
##                   Kappa : 0.6639
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8621
##             Specificity : 0.8000
##          Pos Pred Value : 0.8333
##          Neg Pred Value : 0.8333
##              Prevalence : 0.5370
##          Detection Rate : 0.4630
##    Detection Prevalence : 0.5556
##       Balanced Accuracy : 0.8310
##
##        'Positive' Class : 0
##
```

Method	Accuracy	LowerCI	UpperCI	Sensitivity	Specificity	RMSE	Rsquare
Model 1 - K Means	0.315	0.195	0.456	0.276	0.36	0.561	0.134
Model 2 - PLS	0.815	0.544	0.805	0.724	0.64	0.430	0.394
Model 3 - GLM	0.815	0.686	0.907	0.828	0.80	0.430	0.394
Model 4 - LDA	0.796	0.665	0.894	0.828	0.76	0.451	0.348
Model 5 - QDA	0.778	0.644	0.880	0.759	0.80	0.471	0.310
Model 6 - LOESS	0.833	0.707	0.921	0.862	0.80	0.408	0.441

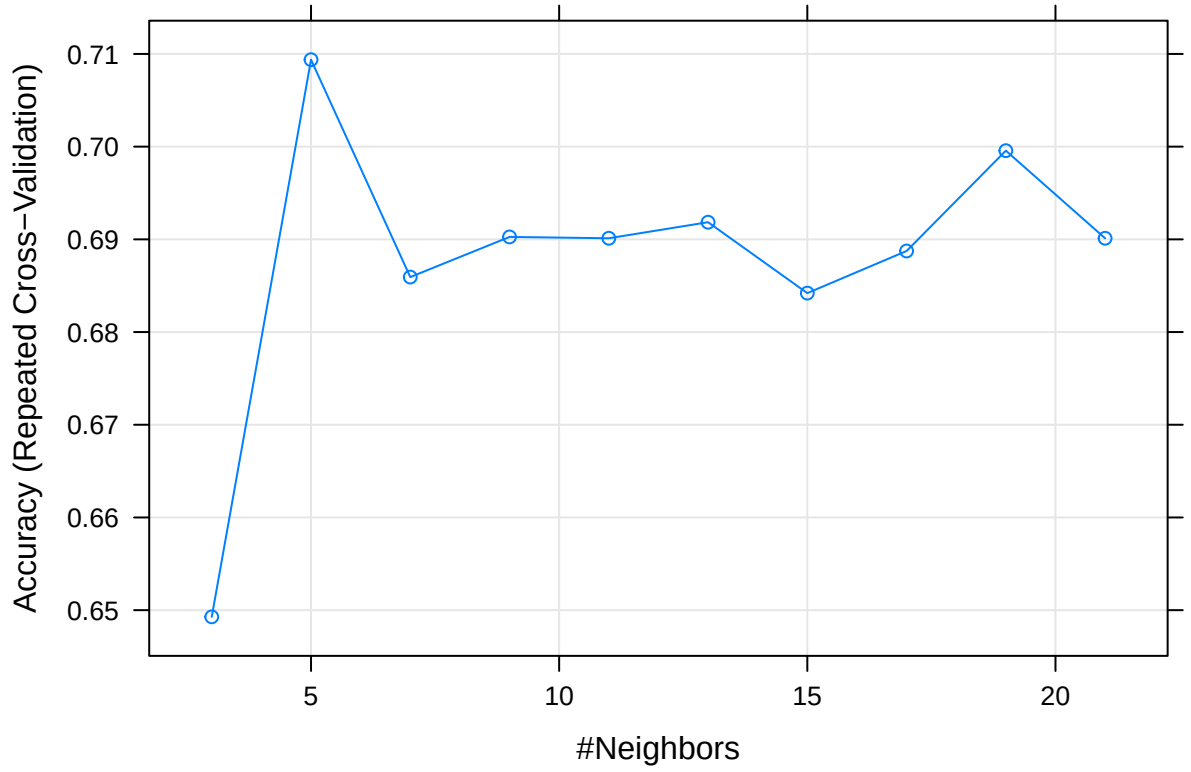
Loess is our best model so far. This implies that our data set has a number of outliers which may be skewing other model outcomes.

Model 7 - K-Nearest Neighbors

K-nearest neighbours will look to find the distances between a query and all examples in the data. It selects the number of examples specified (K) closest to the query. It will then choose the most frequent label as this is a classification outcome.

```
set.seed(1)
train_knn <- train(PresenceValue ~  thalach + ca + oldpeak + age + cp + thal ,
                   data = train_set,
                   method = "knn",
                   trControl = control,
                   tuneGrid = data.frame(k = seq(3, 21, 2)))
knn_preds <- predict(train_knn, test_set)

plot(train_knn)
```




```
# Best K used
bestk <- train_knn$bestTune
```

The above graph shows us that the ideal K used for our model is 5.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
```

```
##          0 21  8
##          1  8 17
##
##                Accuracy : 0.7037
##                  95% CI : (0.5639, 0.8202)
##     No Information Rate : 0.537
##     P-Value [Acc > NIR] : 0.009336
##
##                   Kappa : 0.4041
##
##  Mcnemar's Test P-Value : 1.000000
##
##             Sensitivity : 0.7241
##             Specificity : 0.6800
##          Pos Pred Value : 0.7241
##          Neg Pred Value : 0.6800
##              Prevalence : 0.5370
##          Detection Rate : 0.3889
##    Detection Prevalence : 0.5370
##       Balanced Accuracy : 0.7021
##
##        'Positive' Class : 0
##
```

Method	Accuracy	LowerCI	UpperCI	Sensitivity	Specificity	RMSE	Rsquare
Model 1 - K Means	0.315	0.195	0.456	0.276	0.36	0.561	0.134
Model 2 - PLS	0.815	0.544	0.805	0.724	0.64	0.430	0.394
Model 3 - GLM	0.815	0.686	0.907	0.828	0.80	0.430	0.394
Model 4 - LDA	0.796	0.665	0.894	0.828	0.76	0.451	0.348
Model 5 - QDA	0.778	0.644	0.880	0.759	0.80	0.471	0.310
Model 6 - LOESS	0.833	0.707	0.921	0.862	0.80	0.408	0.441
Model 7 - KNN	0.704	0.564	0.820	0.724	0.68	0.544	0.163

Model 8 - Random Forest

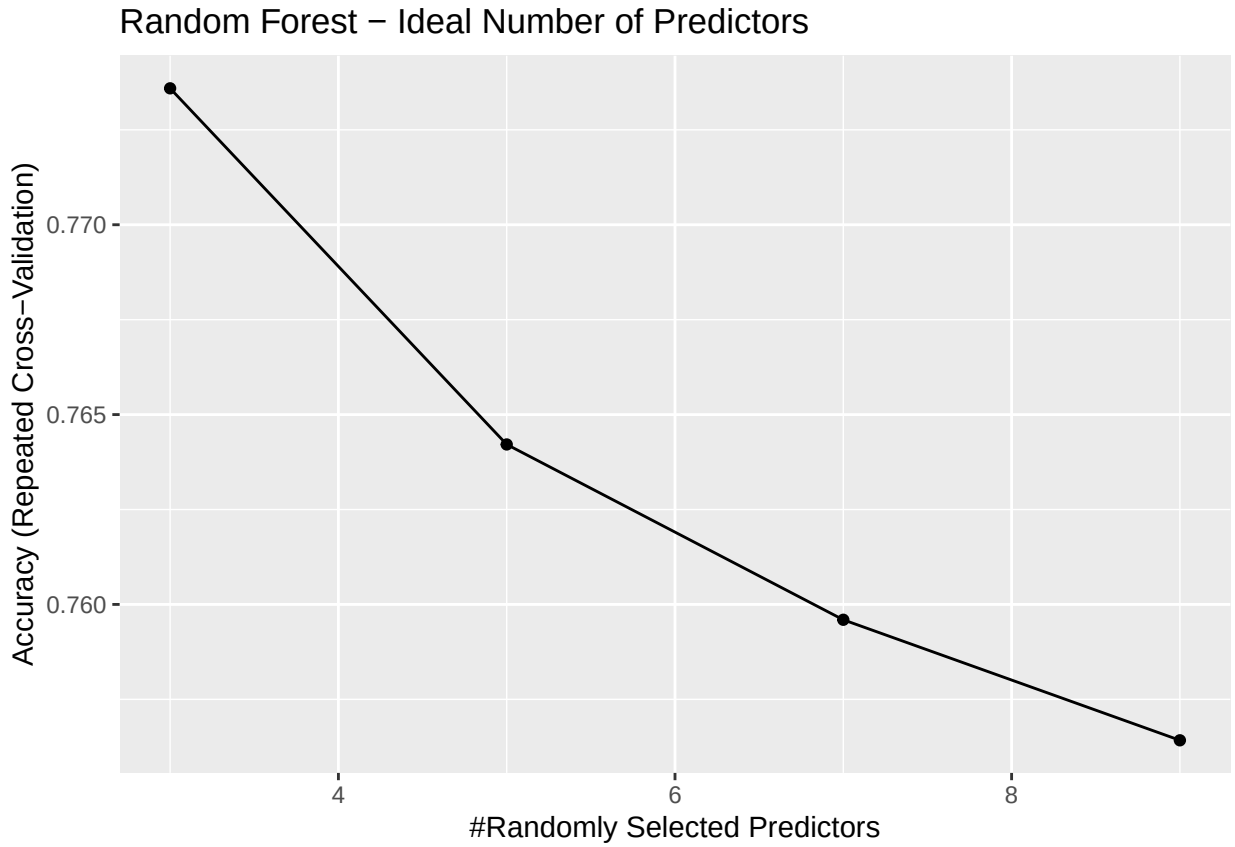
Random forests improve prediction performance and reduce instability by averaging multiple decision trees.

To achieve this firstly it uses aggregation which generates many predictors, each using regression or classification trees, and then forming a final prediction based on the average prediction of all these trees.

Secondly it assures that the individual trees are not the same, by using the bootstrap to induce randomness.

```
set.seed(1)
train_rf <- train(PresenceValue ~  thalach + ca + oldpeak + age + cp + thal ,
                  data = train_set,
                  method = "rf",
                  trControl = control,
                  importance = TRUE,
                  tuneGrid = data.frame(mtry = c(3,5,7,9)))

rf_preds <- predict(train_rf, test_set)
```




```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 23  5
##          1  6 20
##
##                Accuracy : 0.7963
##                  95% CI : (0.6647, 0.8937)
##     No Information Rate : 0.537
##     P-Value [Acc > NIR] : 6.822e-05
##
##                   Kappa : 0.5915
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.7931
##             Specificity : 0.8000
##          Pos Pred Value : 0.8214
##          Neg Pred Value : 0.7692
##              Prevalence : 0.5370
##          Detection Rate : 0.4259
##    Detection Prevalence : 0.5185
##       Balanced Accuracy : 0.7966
##
##        'Positive' Class : 0
```

##

Method	Accuracy	LowerCI	UpperCI	Sensitivity	Specificity	RMSE	Rsquare
Model 1 - K Means	0.315	0.195	0.456	0.276	0.36	0.561	0.134
Model 2 - PLS	0.815	0.544	0.805	0.724	0.64	0.430	0.394
Model 3 - GLM	0.815	0.686	0.907	0.828	0.80	0.430	0.394
Model 4 - LDA	0.796	0.665	0.894	0.828	0.76	0.451	0.348
Model 5 - QDA	0.778	0.644	0.880	0.759	0.80	0.471	0.310
Model 6 - LOESS	0.833	0.707	0.921	0.862	0.80	0.408	0.441
Model 7 - KNN	0.704	0.564	0.820	0.724	0.68	0.544	0.163
Model 8 - Random Forest	0.796	0.665	0.894	0.793	0.80	0.451	0.350

So we have now run 8 different models. We shall test the best ones on the validation set to be sure we have created a working algorithm.

Looking at final accuracies we can see that the best models are Loess, GLM, RF, LDA and QDA.

However, sensitivity in our model is predicting the proportion of patients who do not have heart disease and specificity those who do. In sense of how this data is used these therefore as the most important metrics, as we want to be able to properly diagnose patients who have heart disease and identify those who don't.

Using this logic therefore GLM, Loess and RF are the best models.

Present Modeling Results

Validating the best models

We will now test GLM, Loess and RF against the final validation data set.

Method	Accuracy	LowerCI	UpperCI	Sensitivity	Specificity
Model 3 - GLM	0.867	0.693	0.962	0.875	0.857
Model 5 - Loess	0.867	0.693	0.962	0.875	0.857
Model 8 - Random Forest	0.767	0.577	0.901	0.750	0.786

Validating the best models as an ensemble

Let us create an ensemble validation set using the best three models identified. We will use the ensemble to generate a majority prediction of the presence of heart disease.

```
ensemble <- cbind(glm = glm_preds == "0",
                  loess = loess_preds == "0",
                  rf = rf_preds == "0")

ensemble_preds <- ifelse(rowMeans(ensemble) > 0.5, "0", "1")

Accuracy <- mean(ensemble_preds == validation.y)

confusionmatrix.results <- confusionMatrix(factor(ensemble_preds), factor(validation.y))

# Accuracy <- confusionmatrix.results$overall[1] Alternative way of showing accuracy already calculated
```

```
LowerCI <- confusionmatrix.results$overall[3]
UpperCI <- confusionmatrix.results$overall[4]

Sensitivity <- confusionmatrix.results$byClass[1]
Specificity <- confusionmatrix.results$byClass[2]

# Here we add the results to a table.
Validation_Results <- rbind(Validation_Results,
                            tibble(
                              Method = "Ensemble Final Validation",
                              Accuracy = round(Accuracy, 3),
                              LowerCI = round(LowerCI, 3),
                              UpperCI = round(UpperCI,3),
                              Sensitivity = round(Sensitivity,3),
                              Specificity = round(Specificity,3)              ))


Validation_Results %>% knitr::kable()
```

Method	Accuracy	LowerCI	UpperCI	Sensitivity	Specificity
Model 3 - GLM	0.867	0.693	0.962	0.875	0.857
Model 5 - Loess	0.867	0.693	0.962	0.875	0.857
Model 8 - Random Forest	0.767	0.577	0.901	0.750	0.786
Ensemble Final Validation	0.867	0.693	0.962	0.875	0.857

Discuss Model Performance

After training the data and testing it against the various methods we selected GLM, Loess and RF as the best models. In order to improve on these we created an ensemble which would use a majority rule when determining the binary outcome for the presence of heart disease.

Our final ensemble model has an accuracy of 86.7% which is an improvement on the results achieved by John Gennari's "Models of incremental concept formation" which utilised CLASSIT conceptual clustering of 78.9%.

In addition to this both our final model's sensitivity and specificity are above 85% at 87.5% and 85.7% respectively meaning we are able to determine both the presence and absence of heart disease with relatively similar accuracy.

Conclusion

Summary

GLM, Loess and RF were all effective methods in detecting heart disease and when collated into an ensemble model these improved our detection ability with a final accuracy of 86.7%.

Limitations/Future work

The data sample of ~299 observations is quite small for this sort of analysis, we should ideally increase this. We did attempt to join the four different databases to increase the size however only Cleveland's data has

taken the majority of readings. I wanted to analyse as many variables as possible so reduced the number of observations in favour of variables, this was mainly so I could compare my accuracy with previously completed studies (a good benchmark for improvement).

PCA, PCR, PLS and K-means work better with a higher number of continuous variables, if we could find another database with these then these method outcomes would surely improve. We might also consider using CATPCA to try to improve the use of catergorical predictors.

In terms of how we could improve upon the data we possess. We reduced the number of variables utilised base on visualisation determination. Though we could re run the models with other predictors available such as exang (exercise induced angina), slope (the slope of the peak exercise ST segment) and form of thalassemia.

Another method which could potentially identify useful predictors would be stepwise regression which adds/removes these from the model based on the p-value produced.