# Assignment Part 3

## 1.0 Overview of the problem

The study object of this project is Cardiovascular disease (CVD), which is causing the most death among all the disease in the world. According to the report of World Helath Organization, about 30% of deaths in the world is due to cardiovascular disease. Every year, cardiovascular disease kills about 18 million people. In Australia, about 119 people die from cardiovascular disease each day on average. The good news is that 90% of cardiovascular disease are preventable. Therefore, it is vital to study the factors that might cause cardiovascular disease and take targeted preventive measures.

The aim of this project is to explore the relationship between cardiovascular disease and all the relevant factors in the dataset, and to find ways to prevent CVD or reduce the risk of suffering the disease. In doing so, we will analyze the correlation between CVD and all the relevant factors, and try to build a statistical model to explain the relationships between CVD and the most correlated factors. We will also use the model to predict whether a patient has the condition. Because one patient can only be having the condition or not having the condition, this problem is actually a binary classification problem.

```
data <- read.csv('cardio_train.csv', header = T, sep = ";")
data_gbm_0 = data
library(ggplot2)
library(ggthemes)
library(class)
library(data.table)
library(e1071)
library(Rmisc)
```

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

```
## Loading required package: plyr
```

```
library(ggcorrplot)
library(mlr)
```

```
## Loading required package: ParamHelpers
```

```
##
## Attaching package: 'mlr'
```

```
## The following object is masked from 'package:e1071':
##
##     impute
```

```
library(caret)
```

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

```
## The following object is masked from 'package:mlr':
##
##     train
```

```
library(pROC)
```

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

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

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

```
library(ROCR)
```

```
## Loading required package: gplots
```

```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

```
##
## Attaching package: 'ROCR'
```

```
## The following object is masked from 'package:mlr':
##
##     performance
```

# 2.0 Dataset Description

```
#data dimension
dim(data)
```

```
## [1] 70000    13
```

```
#presence of null values
sum(is.na(data))
```

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

```
head(data)
```

```
##   id   age gender height weight ap_hi ap_lo cholesterol gluc smoke alco
## 1  0 18393      2    168     62   110    80           1    1     0    0
## 2  1 20228      1    156     85   140    90           3    1     0    0
## 3  2 18857      1    165     64   130    70           3    1     0    0
## 4  3 17623      2    169     82   150   100           1    1     0    0
## 5  4 17474      1    156     56   100    60           1    1     0    0
## 6  8 21914      1    151     67   120    80           2    2     0    0
##   active cardio
## 1      1      0
## 2      1      1
## 3      0      1
## 4      1      1
## 5      0      0
## 6      0      0
```

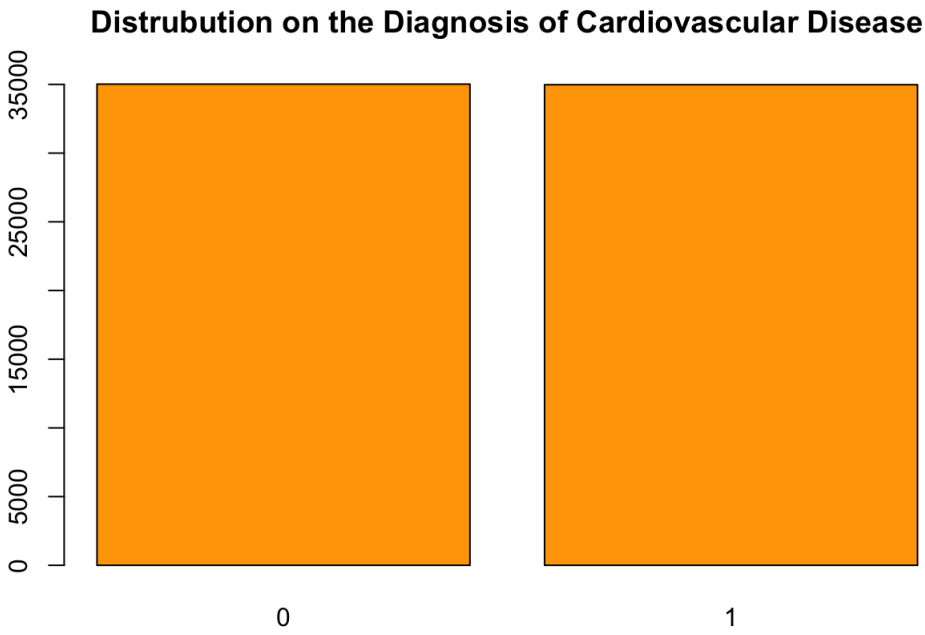
```
summary(data)
```

```
##        id             age            gender         height      
##  Min.   :    0   Min.   :10798   Min.   :1.00   Min.   : 55.0  
##  1st Qu.:25007   1st Qu.:17664   1st Qu.:1.00   1st Qu.:159.0  
##  Median :50002   Median :19703   Median :1.00   Median :165.0  
##  Mean   :49972   Mean   :19469   Mean   :1.35   Mean   :164.4  
##  3rd Qu.:74889   3rd Qu.:21327   3rd Qu.:2.00   3rd Qu.:170.0  
##  Max.   :99999   Max.   :23713   Max.   :2.00   Max.   :250.0  
##      weight           ap_hi             ap_lo           cholesterol   
##  Min.   : 10.00   Min.   : -150.0   Min.   :  -70.00   Min.   :1.000  
##  1st Qu.: 65.00   1st Qu.:  120.0   1st Qu.:   80.00   1st Qu.:1.000  
##  Median : 72.00   Median :  120.0   Median :   80.00   Median :1.000  
##  Mean   : 74.21   Mean   :  128.8   Mean   :   96.63   Mean   :1.367  
##  3rd Qu.: 82.00   3rd Qu.:  140.0   3rd Qu.:   90.00   3rd Qu.:2.000  
##  Max.   :200.00   Max.   :16020.0   Max.   :11000.00   Max.   :3.000  
##       gluc            smoke              alco             active      
##  Min.   :1.000   Min.   :0.00000   Min.   :0.00000   Min.   :0.0000  
##  1st Qu.:1.000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:1.0000  
##  Median :1.000   Median :0.00000   Median :0.00000   Median :1.0000  
##  Mean   :1.226   Mean   :0.08813   Mean   :0.05377   Mean   :0.8037  
##  3rd Qu.:1.000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:1.0000  
##  Max.   :3.000   Max.   :1.00000   Max.   :1.00000   Max.   :1.0000  
##      cardio      
##  Min.   :0.0000  
##  1st Qu.:0.0000  
##  Median :0.0000  
##  Mean   :0.4997  
##  3rd Qu.:1.0000  
##  Max.   :1.0000
```

```
library(mlr)
summarizeColumns(data)
```

```
##           name    type na         mean         disp  median        mad
## 1           id integer  0 4.997242e+04 2.885130e+04 50001.5 36981.9744
## 2          age integer  0 1.946887e+04 2.467252e+03 19703.0  2536.7286
## 3       gender integer  0 1.349571e+00 4.768380e-01     1.0     0.0000
## 4       height integer  0 1.643592e+02 8.210126e+00   165.0     7.4130
## 5       weight numeric  0 7.420569e+01 1.439576e+01    72.0    11.8608
## 6        ap_hi integer  0 1.288173e+02 1.540114e+02   120.0    14.8260
## 7        ap_lo integer  0 9.663041e+01 1.884725e+02    80.0     1.4826
## 8  cholesterol integer  0 1.366871e+00 6.802503e-01     1.0     0.0000
## 9         gluc integer  0 1.226457e+00 5.722703e-01     1.0     0.0000
## 10       smoke integer  0 8.812857e-02 2.834838e-01     0.0     0.0000
## 11        alco integer  0 5.377143e-02 2.255677e-01     0.0     0.0000
## 12      active integer  0 8.037286e-01 3.971791e-01     1.0     0.0000
## 13      cardio integer  0 4.997000e-01 5.000035e-01     0.0     0.0000
##      min   max nlevs
## 1      0 99999     0
## 2  10798 23713     0
## 3      1     2     0
## 4     55   250     0
## 5     10   200     0
## 6   -150 16020     0
## 7    -70 11000     0
## 8      1     3     0
## 9      1     3     0
## 10     0     1     0
## 11     0     1     0
## 12     0     1     0
## 13     0     1     0
```

```
barplot(table(data$cardio),
        main="Distrubution on the Diagnosis of Cardiovascular Disease", col="orange")
```




The cardiovascular disease dataset contains 13 columns and 70,000 patient samples in total, where the 'id' column is a unique and random identifier. The remaining features are: age, gender, height, weight, ap_hi(Systolic blood pressure), ap_lo(Diastolic blood pressure), cholesterol, glucose, smoke, alcohol, active and cardio. The dataset contains both numerical and categorical data.

Among the 13 features, age, height, weight, systolic blood pressure and diastolic blood pressure are numeric features that are also continuous in nature. The remaining features are categorical variables which were already broken into either different levels or binary coded. The target variable of this dataset is the cardio column. It is the diagnosis of cardiovascular disease, which helps us in building classifiers that predict whether a person has presence of cardiovascular disease.The dataset contain no missing values, which means we have a complete set of samples to help classify the presence of cardiovascular disease, making the predictive models have stronger inferential power and reduces biased estimates. The dataset also has a very even distribution on the number of cardiovascular disease diagnosis, with 35,000 samples of patients with cardiovascular disease, and 35,000 samples without.

# 3.0 Data Cleaning

## 3.1 Removing outliers

According to Heart Foundation [1], an optimal blood pressure level is a reading under 120/80 mmHg, in which the first digit represents systolic blood pressure and the second represents diastolic blood pressure. For systolic blood pressure, a reading of 120 - 139 are in the normal to high range, and may consider at risk for high blood pressure. And for diastolic blood pressure, this range sits in between 60 - 90. Blood pressure over 140/90 mmHg is generally considered to be too high, and a person with blood pressure this high is considered to have Hypertension.

According to Narloch and Brandstar[2], the highest blood pressure recorded in an individual was 370/360. Furthermore, according to Sabbah, Anbe, and Stein [3], they have recorded -2 mmHg for diastolic blood pressure. We assume that a systolic blood pressure of 0 means the person is in a serious emergency (heart not pumping blood to the body anymore) and that 0 is a low as it can go.

Therefore, we would consider data less than 0 or more than 370 as outliers of systolic blood pressure (ap_hi), and data less than -2 and more than 360 as outliers of diastolic blood pressure.

```
#removing outliers
data <- data[!(data$ap_hi>370) & !(data$ap_hi<0) ,]
data <- data[!(data$ap_lo>360) & !(data$ap_lo< -2) ,]
dim(data)
```

```
## [1] 69000    13
```

## 3.2 Removing the ID column

```
# drop the id column
data <- data[,-c(1)]
dim(data)
```

```
## [1] 69000    12
```

The id column which contains patient ID was dropped because it was considered uninformative for disease prediction.

```
str(data)
```

```
## 'data.frame':    69000 obs. of  12 variables:
##  $ age        : int  18393 20228 18857 17623 17474 21914 22113 22584 17668 19834 ...
##  $ gender     : int  2 1 1 2 1 1 1 2 1 1 ...
##  $ height     : int  168 156 165 169 156 151 157 178 158 164 ...
##  $ weight     : num  62 85 64 82 56 67 93 95 71 68 ...
##  $ ap_hi      : int  110 140 130 150 100 120 130 130 110 110 ...
##  $ ap_lo      : int  80 90 70 100 60 80 80 90 70 60 ...
##  $ cholesterol: int  1 3 3 1 1 2 3 3 1 1 ...
##  $ gluc       : int  1 1 1 1 1 2 1 3 1 1 ...
##  $ smoke      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ alco       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ active     : int  1 1 0 1 0 0 1 1 1 0 ...
##  $ cardio     : int  0 1 1 1 0 0 0 1 0 0 ...
```

## 3.3 Scaling ‘Age’

```
#changing age in days to years
data$age<-data$age/365
data$age<-round(data$age,digits = 0)

summary(data)
```

```
##       age            gender          height          weight
##  Min.   :30.00   Min.   :1.000   Min.   : 55.0   Min.   : 11.00
##  1st Qu.:48.00   1st Qu.:1.000   1st Qu.:159.0   1st Qu.: 65.00
##  Median :54.00   Median :1.000   Median :165.0   Median : 72.00
##  Mean   :53.32   Mean   :1.349   Mean   :164.4   Mean   : 74.12
##  3rd Qu.:58.00   3rd Qu.:2.000   3rd Qu.:170.0   3rd Qu.: 82.00
##  Max.   :65.00   Max.   :2.000   Max.   :250.0   Max.   :200.00
##      ap_hi            ap_lo         cholesterol         gluc
##  Min.   :  7.0   Min.   :  0.00   Min.   :1.000   Min.   :1.000
##  1st Qu.:120.0   1st Qu.: 80.00   1st Qu.:1.000   1st Qu.:1.000
##  Median :120.0   Median : 80.00   Median :1.000   Median :1.000
##  Mean   :126.3   Mean   : 81.33   Mean   :1.364   Mean   :1.226
##  3rd Qu.:140.0   3rd Qu.: 90.00   3rd Qu.:1.000   3rd Qu.:1.000
##  Max.   :309.0   Max.   :190.00   Max.   :3.000   Max.   :3.000
##      smoke              alco             active           cardio
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:1.0000   1st Qu.:0.0000
##  Median :0.00000   Median :0.00000   Median :1.0000   Median :0.0000
##  Mean   :0.08787   Mean   :0.05359   Mean   :0.8033   Mean   :0.4949
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:1.0000   3rd Qu.:1.0000
##  Max.   :1.00000   Max.   :1.00000   Max.   :1.0000   Max.   :1.0000
```

‘Age’ in days variable was converted into years to bring the scale of the dataset down and for the ease to read.

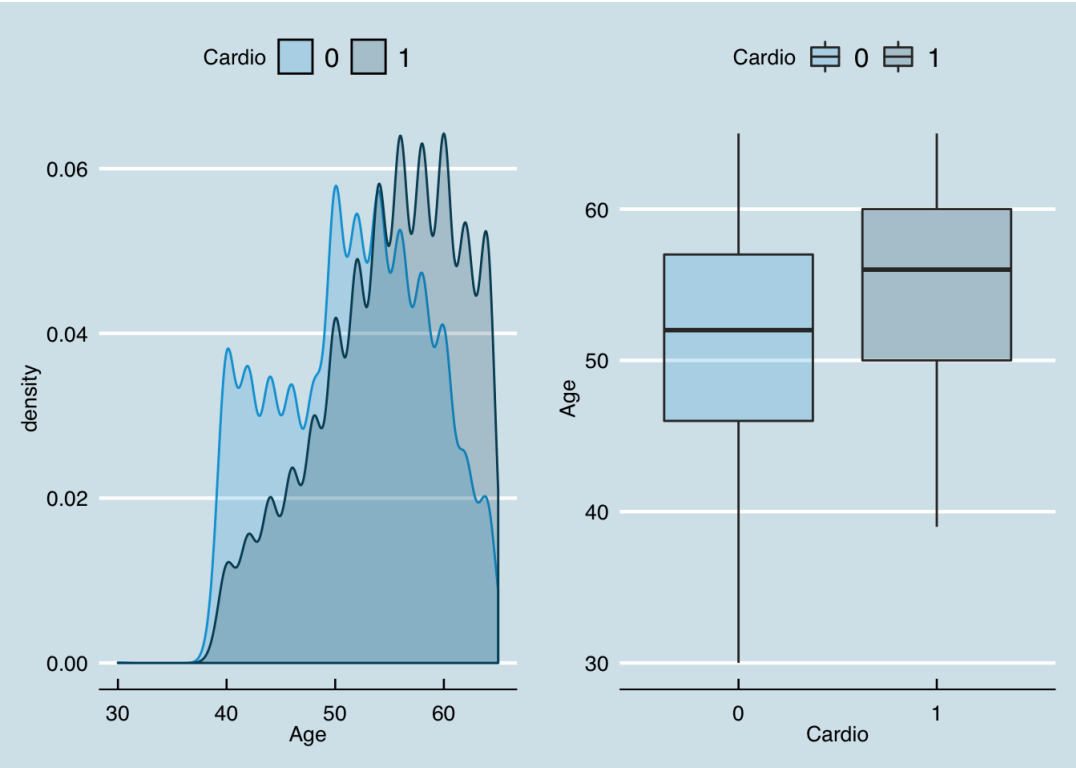
# 4.0 Exploratory Data Analysis

## 4.1 Variable: Age

```
library(plyr)
mu <- ddply(data, "cardio", summarise, grp.mean=mean(age))
head(mu)
```

```
##   cardio grp.mean
## 1      0 51.72172
## 2      1 54.96146
```

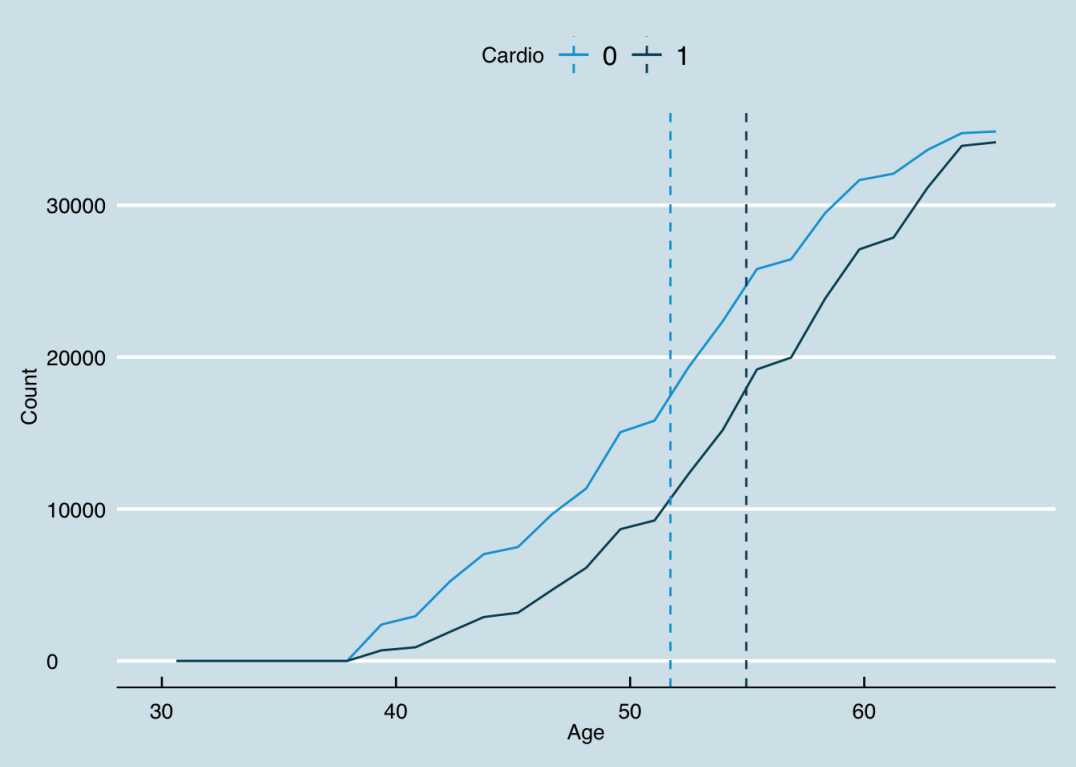
```
age_cardio <- ggplot(data,aes(age,col=as.factor(cardio),fill=as.factor(cardio)))+
  geom_density(alpha=0.2)+
  theme_economist()+
  scale_colour_economist()+
  scale_fill_economist()+
  guides(col=F)+
  labs(fill="Cardio",x="Age")

age_cardio_2 <- ggplot(data,aes(as.factor(cardio),age,fill=as.factor(cardio)))+
  geom_boxplot(alpha=0.2)+
  theme_economist()+
  scale_fill_economist()+
  labs(y="Age",x="Cardio",fill="Cardio")

multiplot(age_cardio,age_cardio_2, cols=2)
```




The graphs above show that the average age of people who have cardiovascular disease (Cardio=1) is higher than people who are healthy. Moreover, the left graph above illustrates the density disparity between the two groups of people and the density line in darker colour seems right-skewed. Hence, we could say that people with cardiovascular disease are likely to be older than healthy people.

In order to take a closer look about the density, we also plotted the cumulative counts of our data.

```
age_cardio <- ggplot(data,aes(x=age, col=as.factor(cardio)))+
  stat_bin(data=subset(data,cardio==0), aes(y=cumsum(..count..)),geom="line", bins = 25)+
  stat_bin(data=subset(data,cardio==1), aes(y=cumsum(..count..)),geom="line", bins = 25)+
  geom_vline(data=mu, aes(xintercept=grp.mean, col=as.factor(cardio)),linetype="dashed")+
  theme_economist()+
  scale_colour_economist()+
  labs(color = "Cardio", x="Age", y="Count")
age_cardio
```



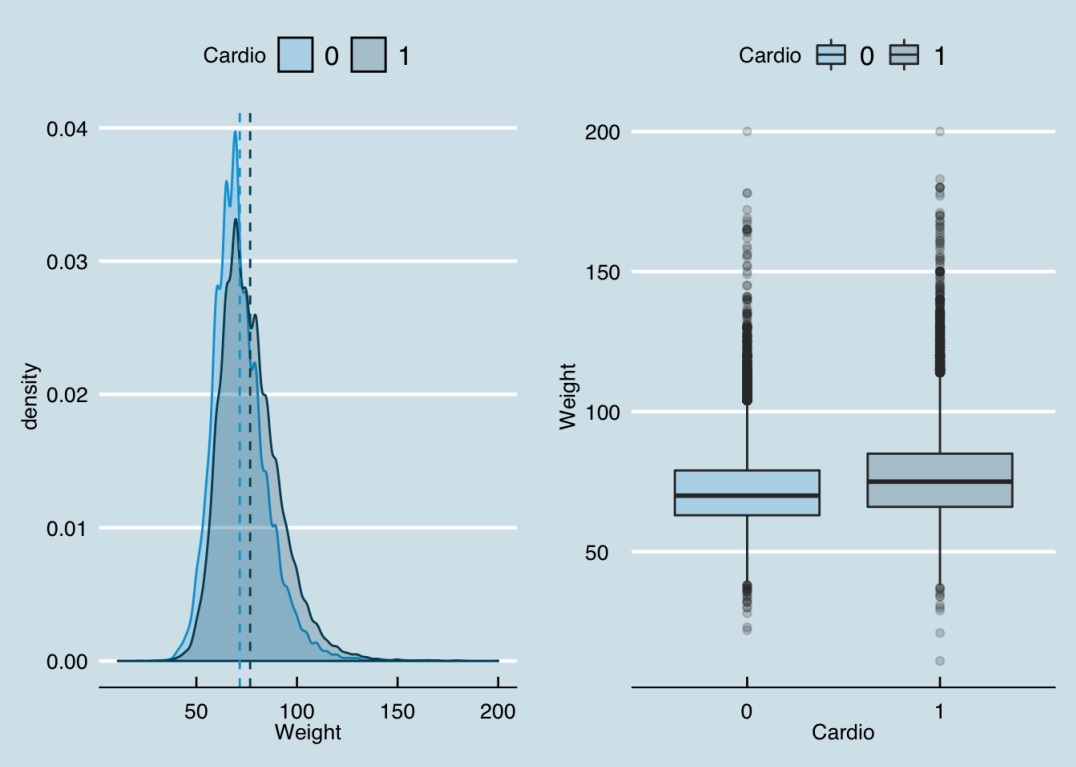

The slope of the graph indicates how rapidly the counts of ill or healthy people change over the age range. More specifically, the slope of people who have cardiovascular disease turned larger as the age is higher, especially within the range where the people's age is higher than the average. By contrast, the slope of healthy people tends to be larger where people are younger.

Because different age ranges show the different rates of ill/healthy people, we could say that age is one of the compelling factors to examine our data in the light of predicting cardiovascular disease.

```
mu <- ddply(data, "cardio", summarise, grp.mean=mean(weight))
head(mu)
```

```
##   cardio grp.mean
## 1      0 71.56541
## 2      1 76.72397
```

```
weight_1 <- ggplot(data,aes(weight,col=as.factor(cardio),fill=as.factor(cardio)))+
  geom_density(alpha=0.2)+
  geom_vline(data=mu, aes(xintercept=grp.mean, col=as.factor(cardio)),linetype="dashed")+
  theme_economist()+
  scale_colour_economist()+
  scale_fill_economist()+
  guides(col=F)+
  labs(fill="Cardio",x="Weight")

weight_2 <- ggplot(data,aes(as.factor(cardio),weight,fill=as.factor(cardio)))+
  geom_boxplot(alpha=0.2)+
  theme_economist()+
  scale_fill_economist()+
  labs(y="Weight",x="Cardio",fill="Cardio")

multiplot(weight_1,weight_2, cols=2)
```

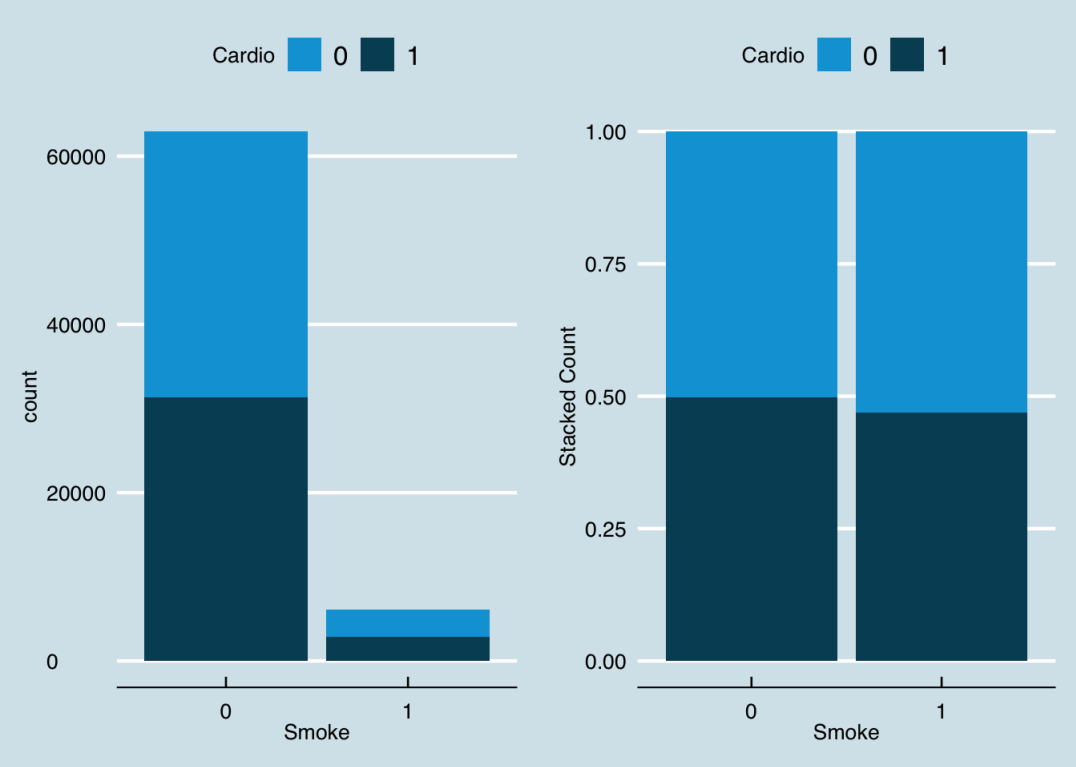



The average weight of healthy people is 71.59kg, whereas the average of ill people is 76.82kg. Besides, while the density plots of the two groups are in a similar shape which is left-skewed, those are distributed slightly different ranges because of their average. Therefore, we could say the weight is apt to be a more significant factor compared to the height.

## 4.2 Variable: Smoke

## 0: non-smoker, 1: smoker

```
smoke_1 <- ggplot(data,aes(as.factor(smoke),fill=as.factor(cardio)))+
  geom_bar(stat="count")+
  theme_economist()+
  scale_fill_economist()+
  labs(x="Smoke",fill="Cardio")

smoke_2 <- ggplot(data,aes(as.factor(smoke),fill=as.factor(cardio)))+
  geom_bar(stat="count",position="fill")+
  theme_economist()+
  scale_fill_economist()+
  labs(x="Smoke",fill="Cardio",y="Stacked Count")

multiplot(smoke_1,smoke_2,cols=2)
```

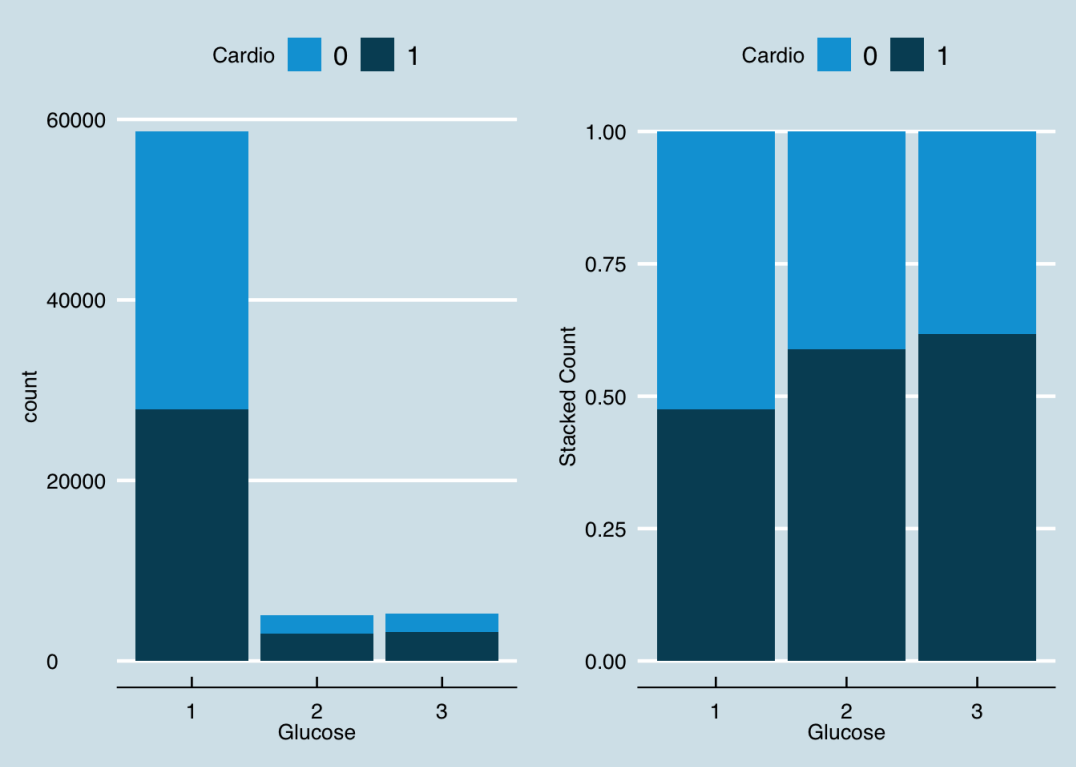



From the graphs shown above, smoking is generally not an influential factor contributing to cardiovascular disease. Evidently, there are far more samples of people who do not smoke in this dataset. The ratio of people that has cardio disease and people that do not are evenly distributed among both smokers and non-smokers. However, smoking was a proven risk for cardiovascular disease in many research papers. Therefore, the reason why we don't see this trend in this dataset may due to the limited samples of smokers.

## 4.3 Variable: Glucose Level

## 1: normal, 2: above normal, 3: well above normal

```
gluc_1 <- ggplot(data,aes(as.factor(gluc),fill=as.factor(cardio)))+
  geom_bar(stat="count")+
  theme_economist()+
  scale_fill_economist()+
  labs(x="Glucose",fill="Cardio")

gluc_2 <- ggplot(data,aes(as.factor(gluc),fill=as.factor(cardio)))+
  geom_bar(stat="count",position="fill")+
  theme_economist()+
  scale_fill_economist()+
  labs(x="Glucose",fill="Cardio",y="Stacked Count")

multiplot(gluc_1,gluc_2,cols=2)
```

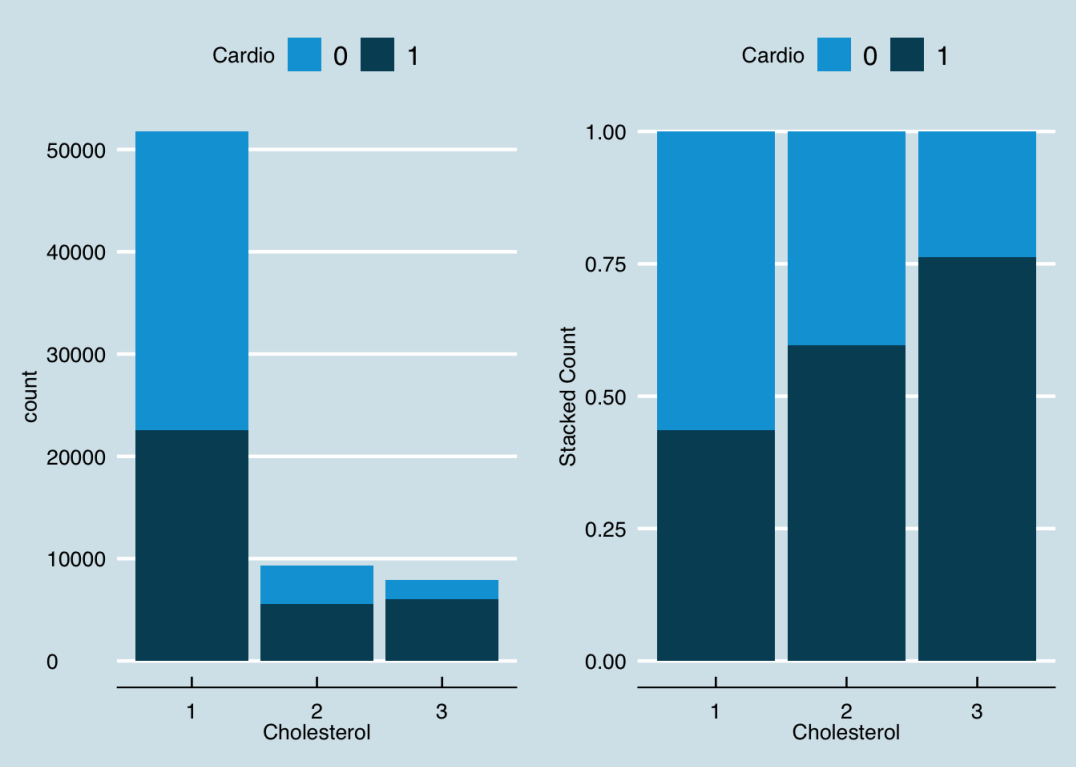



The graph above showed an apparent trend of glucose level contributing to cardiovascular disease. The higher the glucose level, the more likely a person is to have cardio disease.

## 4.4 Variable: Cholesterol

## 1: normal, 2: above normal, 3: well above normal

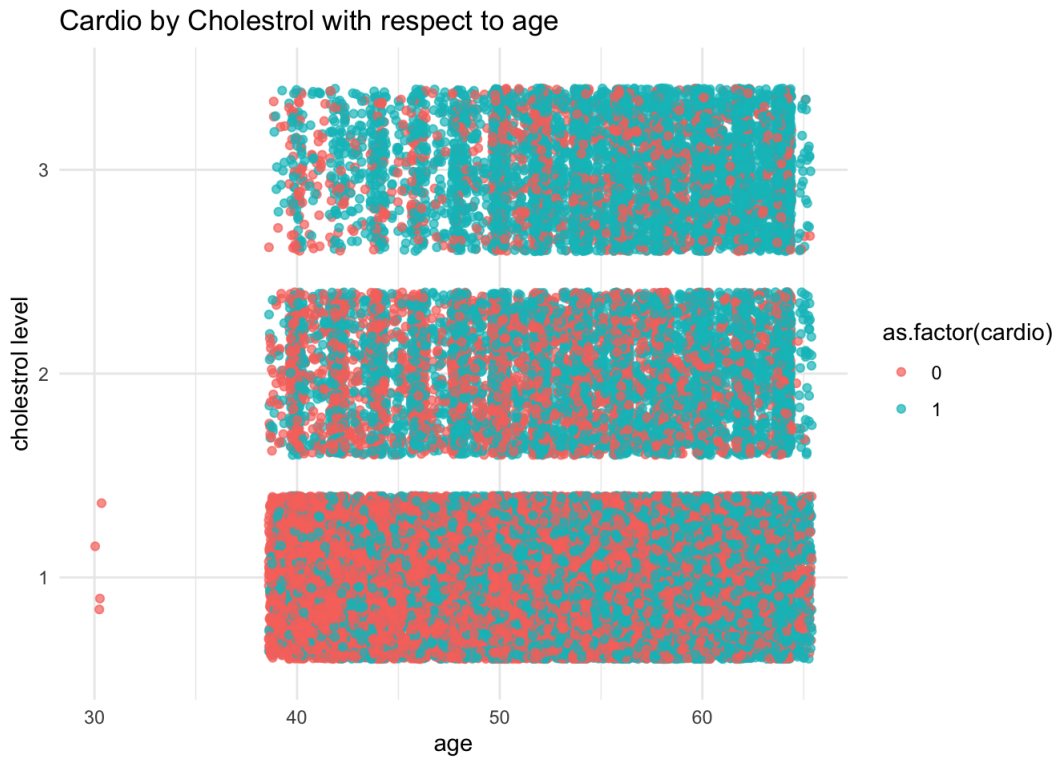
```
chol_1 <- ggplot(data,aes(as.factor(cholesterol),fill=as.factor(cardio)))+
  geom_bar(stat="count")+
  theme_economist()+
  scale_fill_economist()+
  labs(x="Cholesterol",fill="Cardio")

chol_2 <- ggplot(data,aes(as.factor(cholesterol),fill=as.factor(cardio)))+
  geom_bar(stat="count",position="fill")+
  theme_economist()+
  scale_fill_economist()+
  labs(x="Cholesterol",fill="Cardio",y="Stacked Count")

multiplot(chol_1,chol_2,cols=2)
```




Similar to glucose level, from the above graph, we can see a clear trend in cholesterol level and the diagnosis of cardio disease. As cholesterol level increases, the people who are diagnosed with the cardio disease also increases. Therefore, we can say that cholesterol is another factor contributing to cardiovascular disease.

# 4.5 Cardio by Cholestrol with respect to age

```
ggplot(data,
       aes(y = factor(cholesterol,
                      labels = c("1",
                                 "2",
                                 "3")),
           x = age,
           color = as.factor(cardio))) +
  geom_jitter(alpha = 0.7,
              size = 1.5) +
  labs(title = "Cardio by Cholestrol with respect to age",
       x = "age",
       y = "cholestrol level") +
  theme_minimal()
```

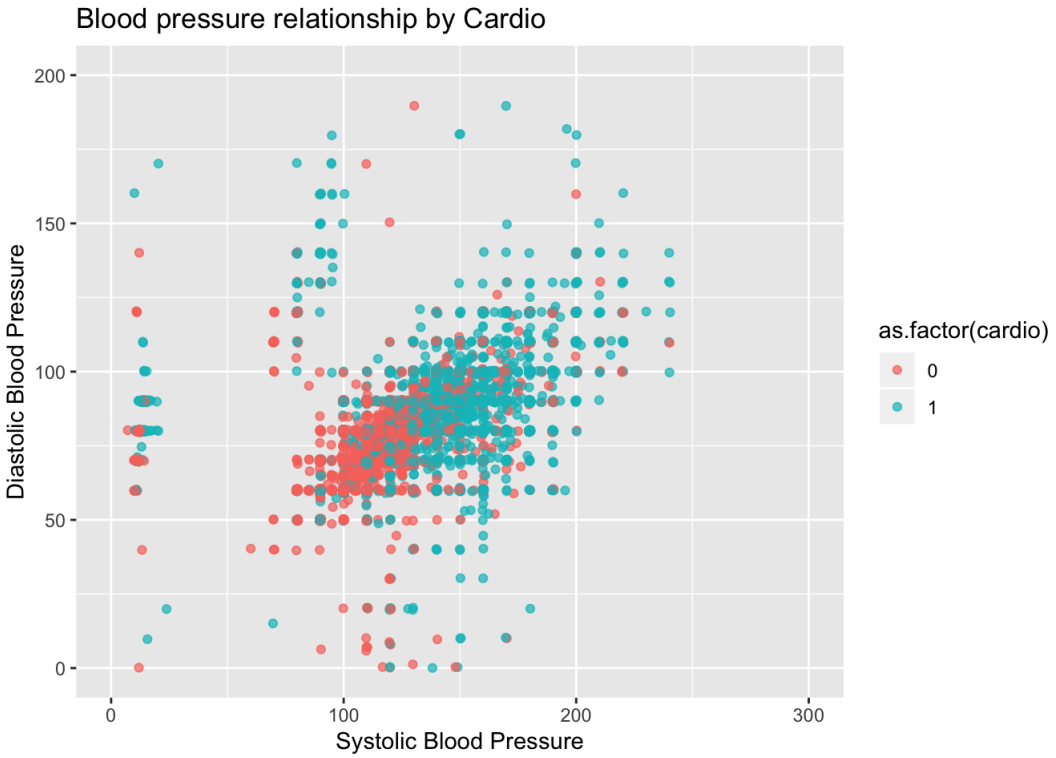



The above scatter graph plots cardio(target) by cholesterol with respect to age. The graph shows that as age and cholesterol level increases, the risk of having cardiovascular disease also increases. We can see that in cholesterol level 1 group, people who have or do not have cardiovascular disease distribute quite evenly, as cholesterol increases, the group of people who have cardiovascular disease dominates. Therefore, we can see that the cholesterol level is strongly linked to cardiovascular disease. In addition, we see that younger people are also less likely to diagnose with cardiovascular disease.

# 4.6 Cardio given by blood pressure (ap_hi and ap_lo)

```
ggplot(data,
       aes(x = ap_hi,
           y = ap_lo,
           color= as.factor(cardio))) +
  geom_jitter(alpha = 0.7,
              size = 1.5) +
  xlim(0, 300)+
  ylim(0, 200)+

  labs(x = "Systolic Blood Pressure",
       y = "Diastolic Blood Pressure",
       title = "Blood pressure relationship by Cardio"
       )
```

```
## Warning: Removed 8 rows containing missing values (geom_point).
```

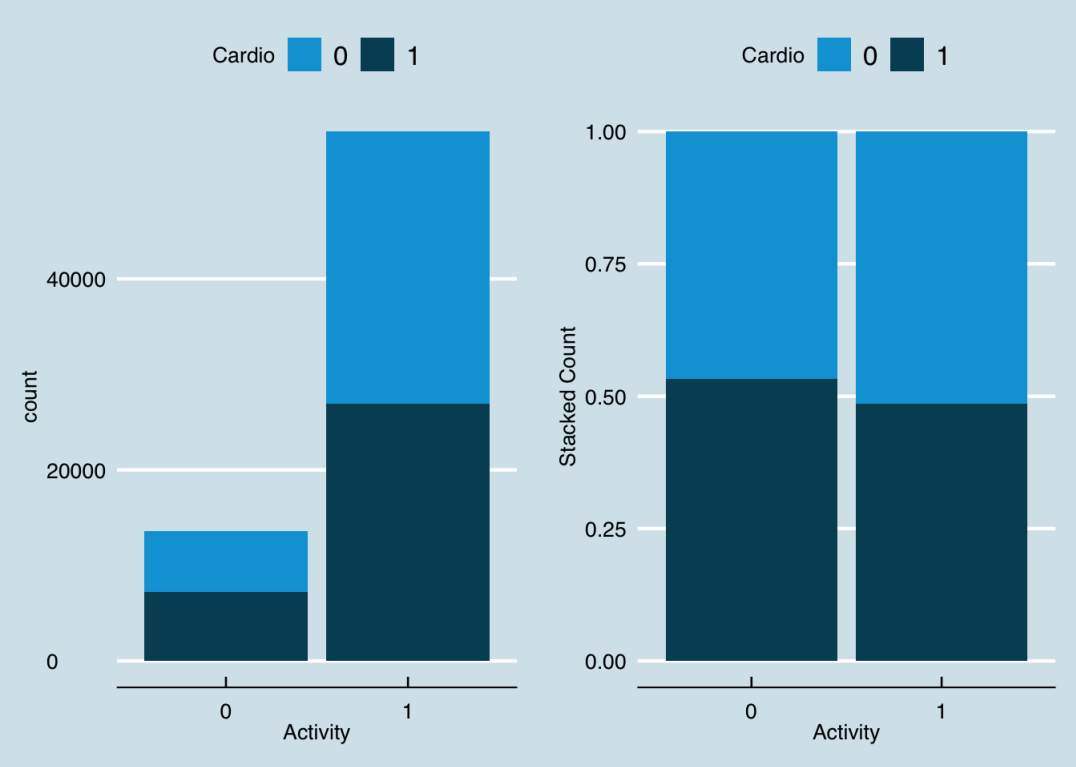



In the above plot, we can observe most of the cardiovascular patients have higher Systolic pressure of more than 130mm/Hg and higher diastolic pressure more than 90mm/Hg.

# 4.7 Variable: Physical Acitivity

## 0: no physical activity, 1: physical activity

```
act_1 <- ggplot(data,aes(as.factor(active),fill=as.factor(cardio)))+
  geom_bar(stat="count")+
  theme_economist()+
  scale_fill_economist()+
  labs(x="Activity",fill="Cardio")

act_2 <- ggplot(data,aes(as.factor(active),fill=as.factor(cardio)))+
  geom_bar(stat="count",position="fill")+
  theme_economist()+
  scale_fill_economist()+
  labs(x="Activity",fill="Cardio",y="Stacked Count")

multiplot(act_1,act_2,cols=2)
```




It is quite a rational finding that we see people who do not exercise have a slightly higher chance of getting cardiovascular disease as we all know that exercise contributes to our health. Again, this can also be seen as a factor that can help us in predicting whether a person is likely to have cardiovascular disease or not.

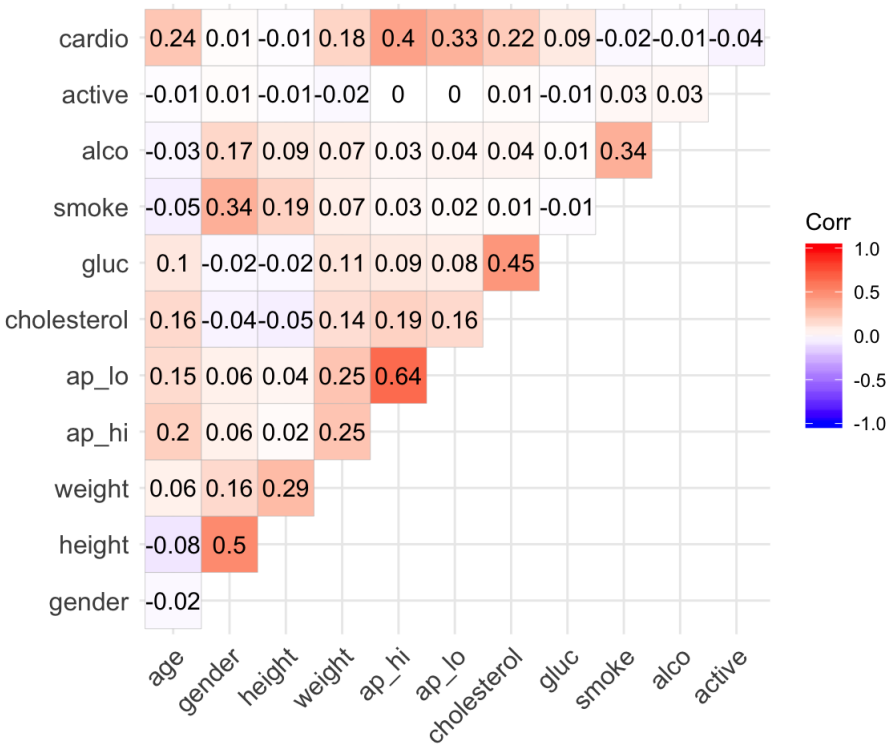
# 4.8 Correlation between variables

```
cor_heart <- round(cor(data[,1:12]),2)
cor_heart
```

```
##                age gender height weight ap_hi ap_lo cholesterol  gluc smoke
## age           1.00  -0.02  -0.08   0.06  0.20  0.15        0.16  0.10 -0.05
## gender       -0.02   1.00   0.50   0.16  0.06  0.06       -0.04 -0.02  0.34
## height       -0.08   0.50   1.00   0.29  0.02  0.04       -0.05 -0.02  0.19
## weight        0.06   0.16   0.29   1.00  0.25  0.25        0.14  0.11  0.07
## ap_hi         0.20   0.06   0.02   0.25  1.00  0.64        0.19  0.09  0.03
## ap_lo         0.15   0.06   0.04   0.25  0.64  1.00        0.16  0.08  0.02
## cholesterol   0.16  -0.04  -0.05   0.14  0.19  0.16        1.00  0.45  0.01
## gluc          0.10  -0.02  -0.02   0.11  0.09  0.08        0.45  1.00 -0.01
## smoke        -0.05   0.34   0.19   0.07  0.03  0.02        0.01 -0.01  1.00
## alco         -0.03   0.17   0.09   0.07  0.03  0.04        0.04  0.01  0.34
## active       -0.01   0.01  -0.01  -0.02  0.00  0.00        0.01 -0.01  0.03
## cardio        0.24   0.01  -0.01   0.18  0.40  0.33        0.22  0.09 -0.02
##                alco active cardio
## age           -0.03  -0.01   0.24
## gender         0.17   0.01   0.01
## height         0.09  -0.01  -0.01
## weight         0.07  -0.02   0.18
## ap_hi          0.03   0.00   0.40
## ap_lo          0.04   0.00   0.33
## cholesterol    0.04   0.01   0.22
## gluc           0.01  -0.01   0.09
## smoke          0.34   0.03  -0.02
## alco           1.00   0.03  -0.01
## active         0.03   1.00  -0.04
## cardio        -0.01  -0.04   1.00
```

```
#gender&height, cholesterol&glucose, alcohol&smoke, gender&smoke
```

```
ggcorrplot(cor_heart, type = "upper", lab = TRUE)
```




As we can see above, cholesterol and glucose have the highest correlation (0.45). It is quite an interesting finding because not much research has proven the link between higher sugar lead to higher cholesterol or vice-versa, but in this case, we see an apparent correlation between these two features. Another strong linkage we can see is between smoke and alcohol(0.34). It's easy to see the rationale behind this correlation as people who smoke may more likely to be alcohol drinkers as well.

# 5.0 Modelling

Many machine learning algorithms are not typically useful when it comes to dealing with categorical data, especially when the category has several levels. In this case, we have chosen the random forest model as it is known to work well with categorical features, when features are on various scales and reduces the problem of over-fitting.

To start with, a random forest model using default parameters was built, and the model will then be fine-tuned by changing 'mtry' and 'ntree', where 'ntree' is the number of trees to grow and 'mtry' is the number of variables randomly sampled as candidates at each split.

```
data$cardio<-as.factor(data$cardio)
```

```
set.seed(123)
subsample <- data[sample(nrow(data),10000),]
dim(subsample)
```

```
## [1] 10000    12
```

We have decided to use a subsample of 10k randomly selected data from the dataset to increase the efficiency of the model.

# Random Forest

```
library(randomForest)
```

```
## randomForest 4.6-14
```

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

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

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

```
set.seed(123)
inTrain <- createDataPartition(subsample$cardio, p=0.6)[[1]]
trainData <- subsample[inTrain,]
testData <-  subsample[-inTrain,]
train_x <- trainData[, -12]
train_y <- trainData$cardio
test_x <- testData[,-12]
test_y <- testData$cardio


training <- data.frame(train_x, target = train_y)
```

```
rf_fit <- randomForest(train_x, train_y, importance = TRUE)

rf_fit
```

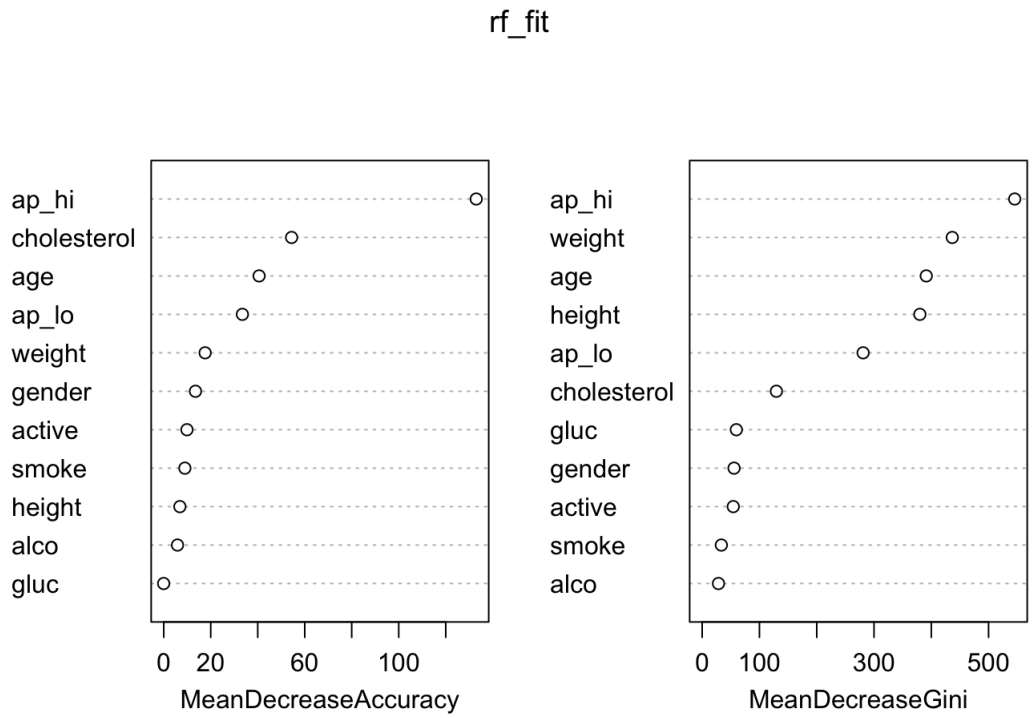
```
##
## Call:
##  randomForest(x = train_x, y = train_y, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 27.1%
## Confusion matrix:
##      0    1 class.error
## 0 2315  697   0.2314077
## 1  929 2060   0.3108063
```

# Feature selection - RF

```
importance(rf_fit, type = 1)
```

```
##             MeanDecreaseAccuracy
## age                    40.57363627
## gender                 13.50182811
## height                  6.87106978
## weight                 17.60788431
## ap_hi                 132.97928783
## ap_lo                  33.46928722
## cholesterol            54.41205586
## gluc                   -0.03487254
## smoke                   8.99184865
## alco                    5.85098051
## active                  9.87896460
```

```
varImpPlot(rf_fit)
```




```
confusionMatrix(predict(rf_fit, test_x), test_y, mode = 'everything')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1512  649
##          1  495 1343
##
##                Accuracy : 0.7139
##                  95% CI : (0.6996, 0.7279)
##     No Information Rate : 0.5019
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4277
##
##  Mcnemar's Test P-Value : 6.081e-06
##
##             Sensitivity : 0.7534
##             Specificity : 0.6742
##          Pos Pred Value : 0.6997
##          Neg Pred Value : 0.7307
##               Precision : 0.6997
##                  Recall : 0.7534
##                      F1 : 0.7255
##              Prevalence : 0.5019
##          Detection Rate : 0.3781
##    Detection Prevalence : 0.5404
##       Balanced Accuracy : 0.7138
##
##        'Positive' Class : 0
##
```

The above summary shows the top important variables and the confusion matrix for this dataset using random forest. Currently, with all features selected, the accuracy in predicting cardiovascular disease is 72.14%. The top 5 features will then be selected to train the model again to see if it will improve model performence. The variables include: ap_hi, ap_lo, age, weight and cholesterol level.

# Fitting model with top 5 features

```
rf_fit2 <- randomForest(cardio ~ age+weight+ap_hi+ap_lo+cholesterol,data = trainData, importance = TRUE)

rf_fit2
```

```
##
## Call:
##  randomForest(formula = cardio ~ age + weight + ap_hi + ap_lo +      cholesterol, data = trainData, impor
tance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 27.76%
## Confusion matrix:
##      0    1 class.error
## 0 2324  688   0.2284197
## 1  978 2011   0.3271997
```

```
feature_rf <- defaultSummary(data.frame(obs = testData$cardio,
                        pred = predict(rf_fit2, testData)))
feature_rf
```

```
##  Accuracy     Kappa
## 0.7039260 0.4076458
```

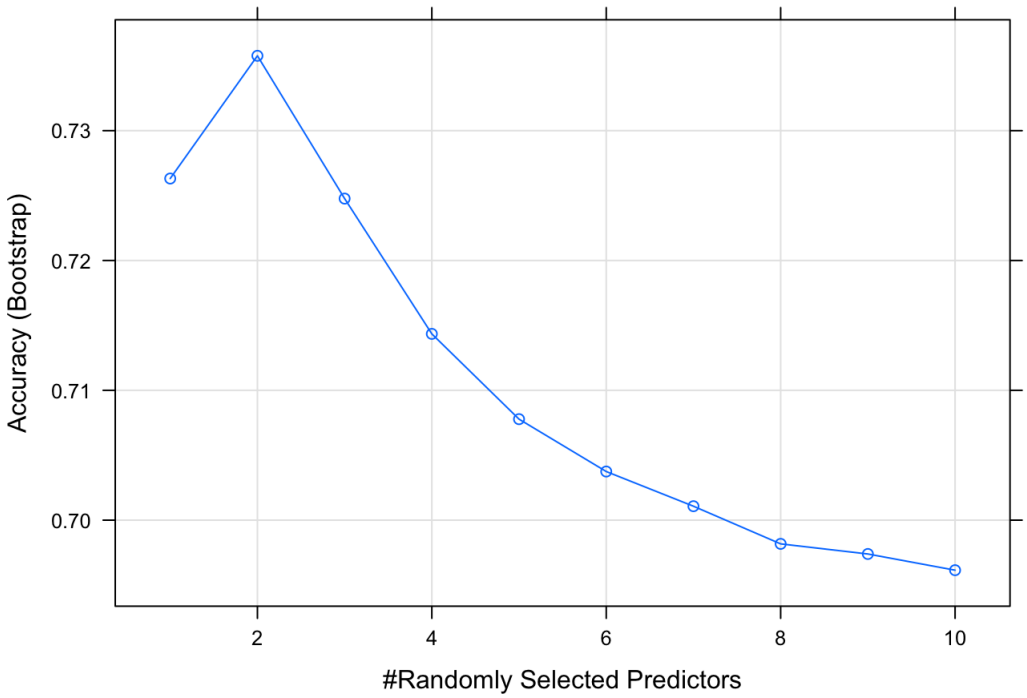
As shown by the result above, selecting the top 5 feature does not improve accuracy, instead, it descreaed the explanatory power of the random forest model. This may be due to the limited features we have in this dataset. Hence in situation where we have limited features to work with, we have decided to include all features to improve the explanatory power of the random forest model.

# Random Forest parameter tuning

```
control <- trainControl(method='cv',
                        number=10,
                        search='grid')

tunegrid <- expand.grid(mtry = (1:10))

rf_gridsearch <- train(cardio ~ .,
                       data = trainData,
                       method = 'rf',
                       metric = 'Accuracy',
                       tuneGrid = tunegrid)

rf_gridsearch
```

```
## Random Forest
##
## 6001 samples
##   11 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 6001, 6001, 6001, 6001, 6001, 6001, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    1    0.7263121  0.4527361
##    2    0.7357661  0.4715636
##    3    0.7247673  0.4495761
##    4    0.7143475  0.4287289
##    5    0.7077792  0.4155886
##    6    0.7037461  0.4075254
##    7    0.7010743  0.4021873
##    8    0.6981733  0.3963833
##    9    0.6973947  0.3948167
##   10    0.6961466  0.3923412
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

```
plot(rf_gridsearch)
```




# Model performance with best tuned parameters

```
confusionMatrix(predict(rf_gridsearch, test_x), test_y, mode ='everything')
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1533  649
##          1  474 1343
##
##                Accuracy : 0.7192
##                  95% CI : (0.705, 0.7331)
##     No Information Rate : 0.5019
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4382
##
##  Mcnemar's Test P-Value : 2.077e-07
##
##             Sensitivity : 0.7638
##             Specificity : 0.6742
##          Pos Pred Value : 0.7026
##          Neg Pred Value : 0.7391
##               Precision : 0.7026
##                  Recall : 0.7638
##                      F1 : 0.7319
##              Prevalence : 0.5019
##          Detection Rate : 0.3833
##    Detection Prevalence : 0.5456
##       Balanced Accuracy : 0.7190
##
##        'Positive' Class : 0
##
```

```
rf.cm <- confusionMatrix(predict(rf_gridsearch, test_x), test_y, mode ='everything')
rf.accuracy <- rf.cm$overall['Accuracy']
rf.sensitivity <- rf.cm$byClass['Sensitivity']
rf.specificity <- rf.cm$byClass['Specificity']
```

By using gridsearch, it was discovered that the optimal 'mtry' is 2 at every split.

# Result evaluation - random forest

As shown by the above confusion matrix, with the best number of mtry selected, we have slightly increased the accuracy from 72.14% to 72.62%. Despite accuracy, all other evaluation metrics such as sensitivity, specificity, precision and F1 score also increased. The final accuracy score for random forest model is 72.62%, which means 72.62% of the test samples were correctly classified into the correct classes.

Among the 5 metrics we proposed to evaluate model performance,the specificity score is relatively lower than all other metrics. Specificity score measures when the actual case is negative (patient has cardiovascular disease), how often is the prediction correct. The lower the score is of concern as specificity score is vitally important for disease detection. Therefore, for future improvement, we should seek to increase model performance in the specificity score.

In contrast, sensitivity score is slightly better than specificity score, which means the model is better at classifying patients without cardiovascular disease.

The random forest model generally runs quite fast after we subsampled the dataset. However, gridsearch did take about 20 minutes to run.

# 5.2 KNN

```
set.seed(1)
inTrain <- createDataPartition(data$cardio, p= 0.6)[[1]]
train <- data[inTrain, -12]
test <- data[-inTrain, -12]
cls.train <- data$cardio[inTrain]
cls.test <- data$cardio[-inTrain]
```

Feature selection was performed using fold change. Here we pick the top 7 features (about half of the number of features in the dataset) and compare the model built by these 7 features with that of using all feautures.

```
library(class)
Train.byClass <- split(train, cls.train)
feature.mean.byClass <- sapply(Train.byClass, colMeans)

# calculate fold change of features by class and take the absolute of its log value
feature.foldChange <- abs(log2(feature.mean.byClass[,1] / feature.mean.byClass[,2]))

# sort the features by fold change
feature.sorted <- sort(feature.foldChange, decreasing=TRUE)

# select the top 7 features
filtered.features1 <- names(feature.sorted)[1:7]
filtered.features1
```

```
## [1] "cholesterol" "smoke"       "ap_hi"       "gluc"        "alco"
## [6] "ap_lo"       "weight"
```

```
set.seed(123)
inTrain <- createDataPartition(subsample$cardio, p=0.6)[[1]]
trainData <- subsample[inTrain,]
testData <-  subsample[-inTrain,]
train <- trainData[, -12]
test <- testData[, -12]
cls.train <- subsample$cardio[inTrain]
cls.test <- subsample$cardio[-inTrain]
```

##kNN classifier

The k-nearest neighbours algorithm (kNN) is a non-parametric classifier. The idea behind kNN is to predict the class of a particular sample taking into account of its nearest neighbour's classes. The sample is classified by the vote of its neighbours, with it being assigned a class which is most common among its k number of neighbours.

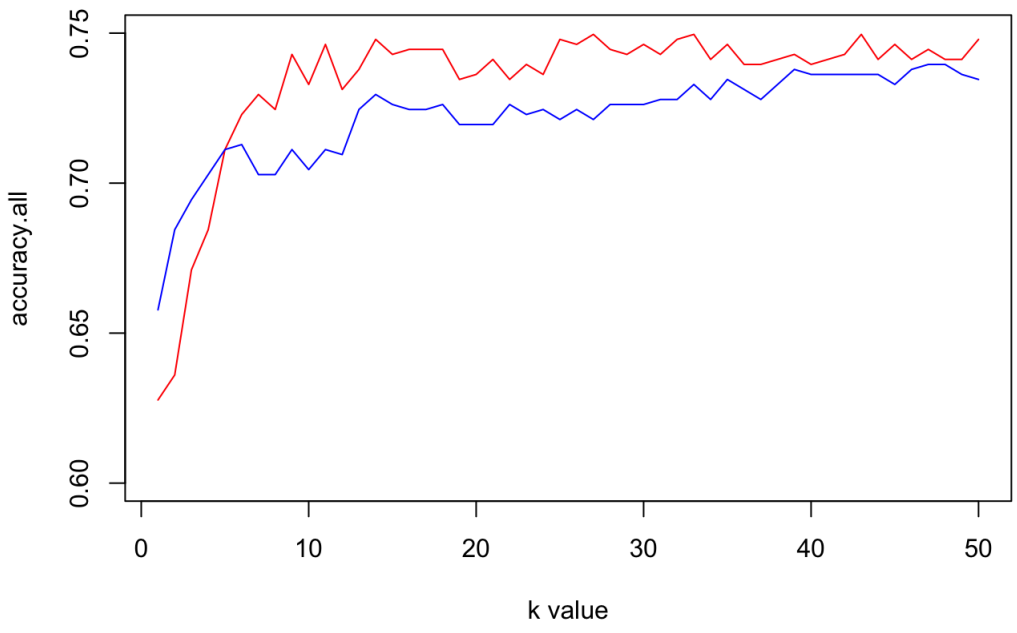
##Parameter selection Here we look at the value of k and from 1 to 50 and determine the best k that maximise specificity while maintaining a reasonable accuracy. As we increases the value of k, the model should become more stable due to majority voting, and thus more likely to make more accurate predictions up to a certain point.

In this project, we have used the top 7 features to build a model in a 10 cross validation and compared that to the model built using all features to decide if we should use the top 7 features subset. We will then decide the optimal number k by looking at the point where the performance of the model no longer benefited by increasing value of k.

# Model tuning using the training Set - maximising accuracy

We plot the accuracy of the model using 10 fold cross validation with k ranging from 1 to 50. The resulting accuracies of the kNN model built by all features are compared with that built by top 7 features. By looking at the plot and the mean accuracies of these two model, it is evident that by using the top 7 features, the model accuracy improves slightly compared to the model built by all features.

```
fold <- createFolds(cls.train, k=10)
knn.acc.k <- c()

#accuracies for all features
accuracy.all = sapply(1:50, function(x){
  knn.correct <- 0
  for (i in 1:length(fold)){
    knn.pred <- knn(train = train[-fold[[i]],], test = train[fold[[i]],], cl = cls.train[-fold[[i]]], k=x)
    cm <- confusionMatrix(table(knn.pred, cls.train[fold[[i]]]))
    knn.accuracy <- cm$overall['Accuracy']
}

knn.acc.k <- c(knn.acc.k, knn.accuracy)
})

#Accuracies for top 8 selected features
accuracy.selectedFeatues = sapply(1:50, function(x){
  knn.correct <- 0
  for (i in 1:length(fold)){
    knn.pred <- knn(train = train[-fold[[i]],filtered.features1], test = train[fold[[i]],filtered.features
1], cl = cls.train[-fold[[i]]], k=x)
    cm <- confusionMatrix(table(knn.pred, cls.train[fold[[i]]]))
    knn.accuracy <- cm$overall['Accuracy']
}

knn.acc.k <- c(knn.acc.k, knn.accuracy)
})

plot(x = 1:50, y=accuracy.all, col = 'red', type = 'l', ylim = c(0.6, 0.75), xlab = 'k value')
points(x = 1:50, y=accuracy.selectedFeatues, col = 'blue', type = 'l', xlab = 'k value')
```




```
mean(accuracy.all)
```

```
## [1] 0.733823
```

```
mean(accuracy.selectedFeatues)
```

```
## [1] 0.7224374
```

##Model tuning using training set - maximising sensitivity

Using the top 7 features to build the model slightly improves the sensitivity compared to the model with all features. However since we want to maximise specificity in this project, we would stick with building the model with all features.

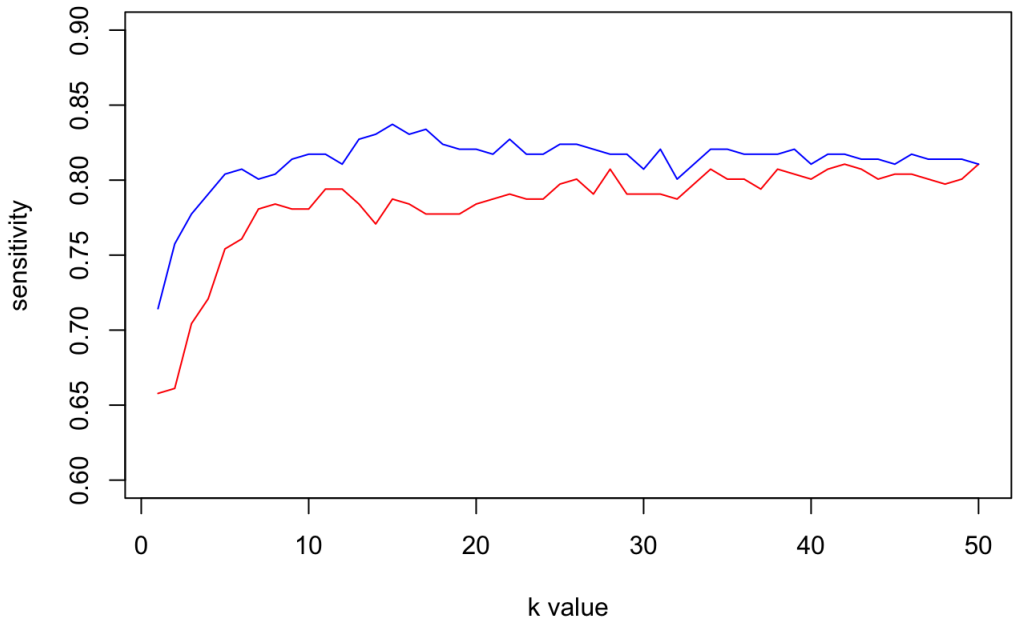
```
knn.sen.k <- c()

#Sensitivity of the model using the selected features
sensitivity.selectedFeatures = sapply(1:50, function(x){
  knn.correct <- 0
  for (i in 1:length(fold)){
    knn.pred <- knn(train = train[-fold[[i]],filtered.features1], test = train[fold[[i]],filtered.features
1], cl = cls.train[-fold[[i]]], k=x)
    conf_mat <- table(knn.pred, cls.train[fold[[i]]])
    knn.sensitivity <- sensitivity(conf_mat)
}

knn.sen.k <- c(knn.sen.k, knn.sensitivity)
})

#Sensitivity of the model using all features
sensitivity.all = sapply(1:50, function(x){
  knn.correct <- 0
  for (i in 1:length(fold)){
    knn.pred <- knn(train = train[-fold[[i]],], test = train[fold[[i]],], cl = cls.train[-fold[[i]]], k=x)
    conf_mat <- table(knn.pred, cls.train[fold[[i]]])
    knn.sensitivity <- sensitivity(conf_mat)
}

knn.sen.k <- c(knn.sen.k, knn.sensitivity)
})

mean(sensitivity.all)
```

```
## [1] 0.783588
```

```
mean(sensitivity.selectedFeatures)
```

```
## [1] 0.8125581
```

```
plot(x = 1:50, y=sensitivity.all, col = "red", type = 'l', ylab = "sensitivity", ylim = c(0.6,0.9), xlab =
'k value')
points(x = 1:50, y=sensitivity.selectedFeatures, col = "blue", type = 'l' )
```




##Model tuning using training set - maximising specificity

As seen in the below graph, using the top 7 features to build the model results a slightly lower specificity compared to all features. Since the objective is to maximise the specificity, the final model should be built by using all features

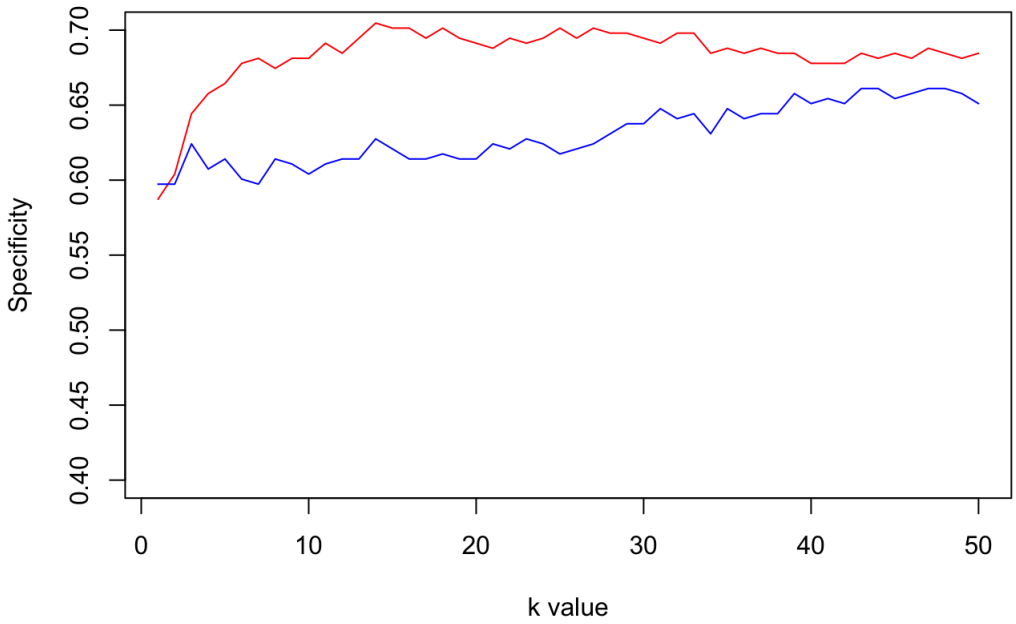
```
knn.spe.k <- c()

specificity.all = sapply(1:50, function(x){
  knn.correct <- 0
  for (i in 1:length(fold)){
    knn.pred <- knn(train = train[-fold[[i]],], test = train[fold[[i]],], cl = cls.train[-fold[[i]]], k=x)
    conf_mat <- table(knn.pred, cls.train[fold[[i]]])
    knn.specificity <- specificity(conf_mat)
}

knn.spe.k <- c(knn.spe.k, knn.specificity)
})

specificity.selectedFeatures = sapply(1:50, function(x){
  knn.correct <- 0
  for (i in 1:length(fold)){
    knn.pred <- knn(train = train[-fold[[i]],filtered.features1], test = train[fold[[i]],filtered.features
1], cl = cls.train[-fold[[i]]], k=x)
    conf_mat <- table(knn.pred, cls.train[fold[[i]]])
    knn.specificity <- specificity(conf_mat)
}

knn.spe.k <- c(knn.spe.k, knn.specificity)
})

mean(specificity.all)
```

```
## [1] 0.6834899
```

```
mean(specificity.selectedFeatures)
```

```
## [1] 0.6302685
```

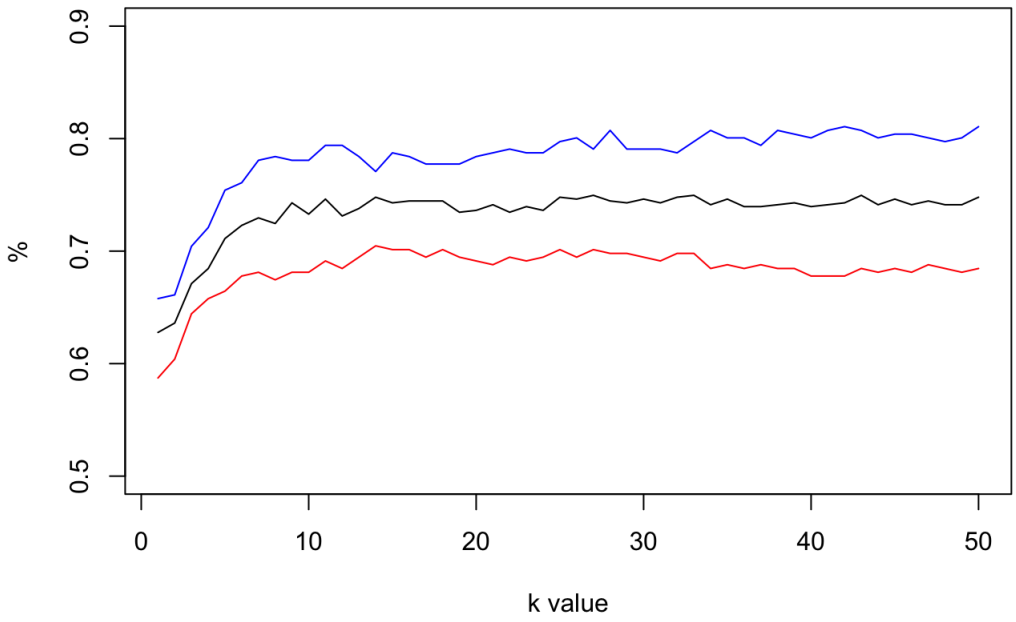
```
plot(x = 1:50, y=specificity.all, col = "red", type = 'l', ylab = "Specificity", ylim = c(0.40, 0.7), xlab =
'k value')
points(x = 1:50, y=specificity.selectedFeatures, col = "blue", type = 'l')
```




# Plot all metrics together to determine the optimal k value for model built by all features.

According to the below graph, the metrics improves as k increases, but specificity peaks at k=19. In case where we are taking a majority vote among labels, we will choose an odd number to have a tiebreaker. Therefore we will use k=19 to build the final model.

```
plot(x = 1:50, y=sensitivity.all, ylim = c(0.5, 0.9), col = "blue", type = 'l', ylab = "%", xlab = 'k value'
)
points(x = 1:50, y=accuracy.all, col = 'black', type = 'l')
points(x = 1:50, y=specificity.all, col = 'red', type = 'l')
```




# Evaluation

Taking all the above into consideration, we have decided to build the model using all features and k=19 and we would evaluate the model using the testing set. As seen below the accuracy is 72% with a confience interval from 71% to 73%, sensitivity is 77% and specificity is 67%. This indicates that the kNN model performs well in reducing false negative but not so much with false positive. We will use this final kNN model to compare with other models to determine the final model we would build the classifier from.

```
knn.pred <- knn(train = train, test = test, cl = cls.train, k=19)

confusionMatrix(table(knn.pred, cls.test))
```

```
## Confusion Matrix and Statistics
##
##          cls.test
## knn.pred    0    1
##        0 1515  686
##        1  492 1306
##
##                Accuracy : 0.7054
##                  95% CI : (0.691, 0.7195)
##     No Information Rate : 0.5019
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4106
##
##  Mcnemar's Test P-Value : 1.874e-08
##
##             Sensitivity : 0.7549
##             Specificity : 0.6556
##          Pos Pred Value : 0.6883
##          Neg Pred Value : 0.7264
##              Prevalence : 0.5019
##          Detection Rate : 0.3788
##    Detection Prevalence : 0.5504
##       Balanced Accuracy : 0.7052
##
##        'Positive' Class : 0
##
```

```
knn.cm <- confusionMatrix(table(knn.pred, cls.test))

knn.accuracy <- knn.cm$overall['Accuracy']
knn.sensitivity <- knn.cm$byClass['Sensitivity']
knn.specificity <- knn.cm$byClass['Specificity']
```

# 5.2 Logistic Regression

Logistic Regression is a statistical method using the logistic function to categorise two possible binary variables such as pass/fail and positive/negative. Logistic Regression simply models the probability of classes and classify the data points. And of course, the probability prediction we have calculated here must be transformed into 0 or 1 at the end.

```
set.seed(123)
subsample <- data[sample(nrow(data),10000),]
dim(subsample)
```

```
## [1] 10000    12
```

```
set.seed(123)
inTrain <- createDataPartition(subsample$cardio, p=0.6)[[1]]
trainData <- subsample[inTrain,]
testData <-  subsample[-inTrain,]
```

```
set.seed(1)
fold <- createFolds(trainData$cardio, k=10)
accuracy <- c()
for(i in 1:length(fold)){
  set.seed(1)
  lr_model <- glm(cardio~., data = trainData[-fold[[i]],], family = "binomial")
  lr_pred <- predict(lr_model, newdata = trainData[fold[[i]],], type = "response")
  lr_pred_num <- ifelse(lr_pred > 0.5, 1, 0)
  accuracy <- c(accuracy, (mean(lr_pred_num == trainData[fold[[i]],]$cardio))*100)
}
k_index <- which.max(accuracy)
```

```
set.seed(1)
lr_model <- glm(cardio~., data = trainData[-fold[[k_index]],], family = "binomial")
lr_pred <- predict(lr_model, newdata = testData, type = "response")
lr_pred_num <- ifelse(lr_pred > 0.5, 1, 0)

table(round(lr_pred_num),testData$cardio)
```

```
##
##         0    1
##    0 1498  657
##    1  509 1335
```

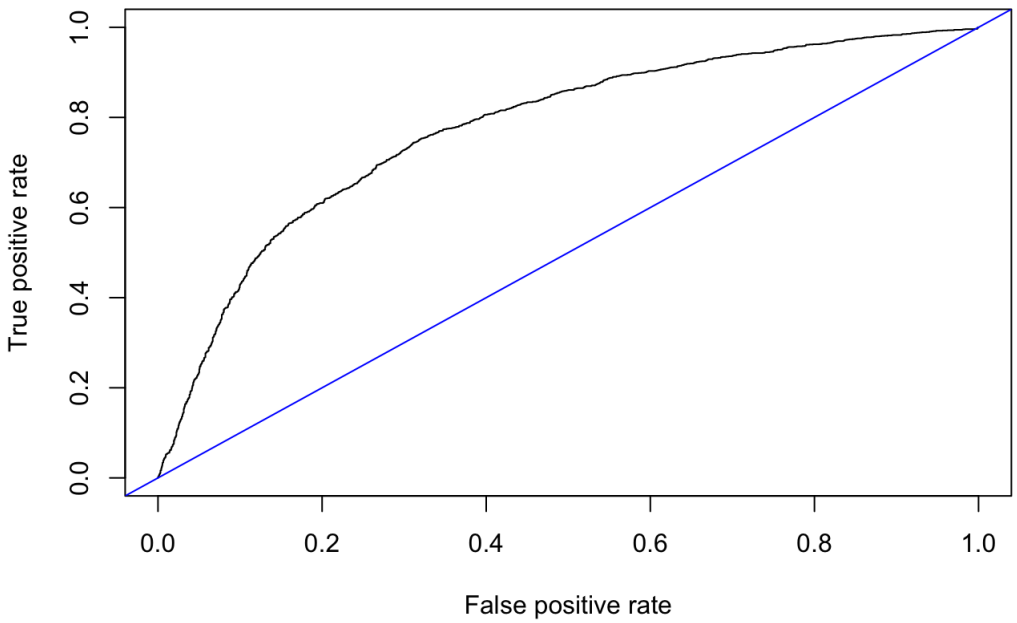
```
paste("Accuracy : ", (mean(lr_pred_num == testData$cardio))*100, "%")
```

```
## [1] "Accuracy :  70.8427106776694 %"
```

We found the best logistic regression model has the highest accuracy based on the result of cross validation and applied that model for the test data. Here, we printed the model evaluation matrix and the accuracy of the prediction like above.

# ROC curve

```
pr <- prediction(lr_pred, testData$cardio)
prf <- ROCR::performance(pr, measure = "tpr", x.measure = "fpr")
plot(prf)
abline(0,1, col="blue")
```



```
auc <- ROCR::performance(pr, measure = "auc")
auc <- auc@y.values[[1]]
paste("auc : ", auc)
```

```
## [1] "auc :  0.775366413336461"
```

```
library(caret)
lr_pred_fac <- as.factor(lr_pred_num)
lr_test_fac <- as.factor(testData$cardio)
def_matrix <- caret::confusionMatrix(lr_pred_fac, lr_test_fac, positive="0", mode="everything")
def_matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1498  657
##          1  509 1335
##
##                Accuracy : 0.7084
##                  95% CI : (0.6941, 0.7225)
##     No Information Rate : 0.5019
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4167
##
##  Mcnemar's Test P-Value : 1.67e-05
##
##             Sensitivity : 0.7464
##             Specificity : 0.6702
##          Pos Pred Value : 0.6951
##          Neg Pred Value : 0.7240
##               Precision : 0.6951
##                  Recall : 0.7464
##                      F1 : 0.7198
##              Prevalence : 0.5019
##          Detection Rate : 0.3746
##    Detection Prevalence : 0.5389
##       Balanced Accuracy : 0.7083
##
##        'Positive' Class : 0
##
```

# Finding the optimal cut-off for LR model

People basically set 0.5 as a cutoff but this could vary depends on how data is distributed in the data set. To improve the model, we computed the below steps to find out the optimal cutoff. Here, we used the training set and the validation set we found as optimal beforehand.

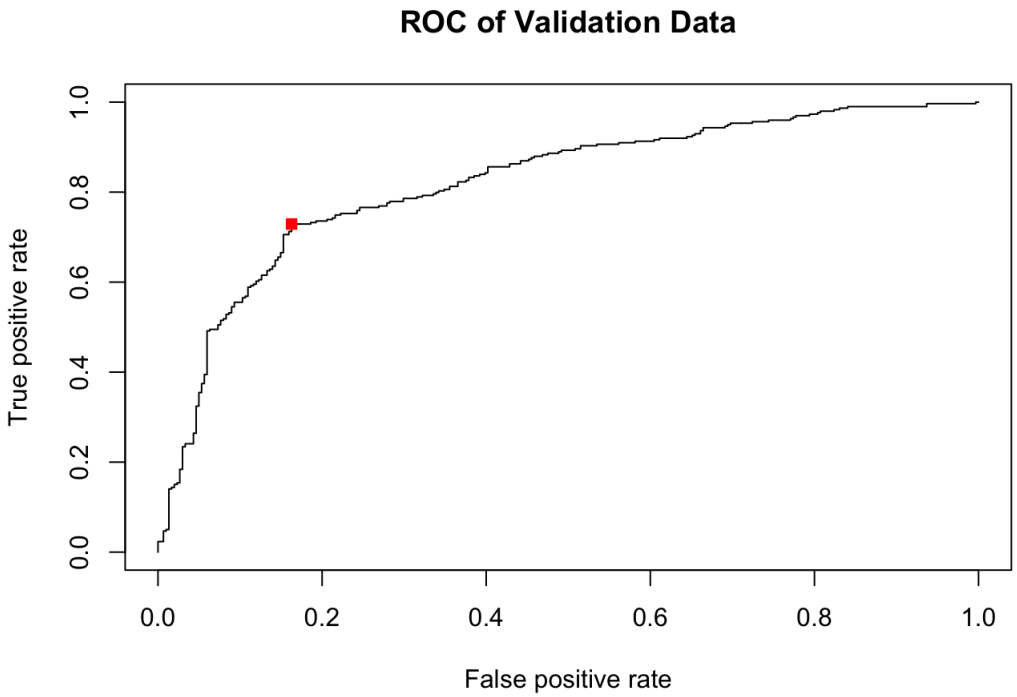
```
library(ROCR)
library("InformationValue")
```

```
##
## Attaching package: 'InformationValue'
```

```
## The following objects are masked from 'package:caret':
##
##     confusionMatrix, precision, sensitivity, specificity
```

```
library(pROC)
```

```
roc_vali <- trainData[fold[[k_index]],]
roc_train <- trainData[-fold[[k_index]],]
set.seed(1)
lr_cut_model <- glm(cardio~., data = roc_train, family = "binomial")
lr_cut_pred <- predict(lr_cut_model, newdata = roc_vali, type = "response")

cut_pr <- prediction(lr_cut_pred, roc_vali$cardio)
cut_prf <- ROCR::performance(cut_pr, measure = "tpr", x.measure = "fpr")
plot(cut_prf, main='ROC of Validation Data')

optid<-(1:length(cut_prf@y.values[[1]][-1]))[((cut_prf@x.values[[1]][-1])^2 + (1-cut_prf@y.values[[1]][-11])
^2)
                                             == min((cut_prf@x.values[[1]][-1])^2 + (1-cut_prf@y.values[[1]]
[-1])^2)]
points(cut_prf@x.values[[1]][-1][optid], cut_prf@y.values[[1]][-1][optid], col='red', pch=15)
```




A red square dot on the graph above denotes the point where it is the shortest to (0, 1) and this means the optimal cut we desired. We would say that this optimal cut point is not the absolute standard, but it is optimal for our data. As we can see, the optimal cut below is not 0.5 which is a default value. Since our data is fairly evenly distributed, this value is slightly different to 0.5, but this could vary much more when the data is biased or skewed, on the other hand

```
optcut<-cut_prf@alpha.values[[1]][-1][optid]

paste("Optimal cut : ", optcut)
```

```
## [1] "Optimal cut :  0.500243063752725"
```

```
set.seed(1)
lr_cut_pred_result <- predict(lr_cut_model, newdata = testData, type = "response")
lr_cut_pred_result_num <- ifelse(lr_cut_pred_result > optcut, 1, 0)
```

# LR with optimal cut-off

```
lr_pred_fac <- as.factor(lr_cut_pred_result_num)
lr_test_fac <- as.factor(testData$cardio)
opt_matrix <- caret::confusionMatrix(lr_pred_fac, lr_test_fac, positive="0", mode="everything")
opt_matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1498  657
##          1  509 1335
##
##                Accuracy : 0.7084
##                  95% CI : (0.6941, 0.7225)
##     No Information Rate : 0.5019
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4167
##
##  Mcnemar's Test P-Value : 1.67e-05
##
##             Sensitivity : 0.7464
##             Specificity : 0.6702
##          Pos Pred Value : 0.6951
##          Neg Pred Value : 0.7240
##               Precision : 0.6951
##                  Recall : 0.7464
##                      F1 : 0.7198
##              Prevalence : 0.5019
##          Detection Rate : 0.3746
##    Detection Prevalence : 0.5389
##       Balanced Accuracy : 0.7083
##
##        'Positive' Class : 0
##
```

```
lr_matrix <- caret::confusionMatrix(lr_pred_fac, lr_test_fac, positive="0", mode="everything")
```

# LR : Compare two results with different cutoff values

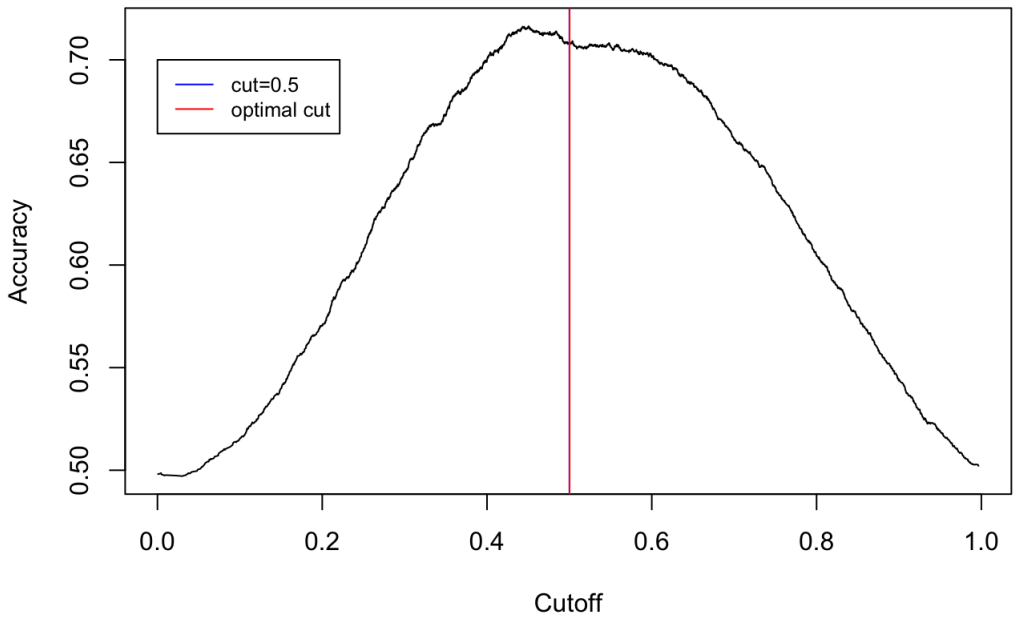
```
Label <- c("Default cutoff", "Optimal cutoff")
Accuracy <- c(def_matrix$overall['Accuracy'], opt_matrix$overall['Accuracy'])
Sensitivity <- c(def_matrix$byClass['Sensitivity'], opt_matrix$byClass['Sensitivity'])
Specificity <- c(def_matrix$byClass['Specificity'], opt_matrix$byClass['Specificity'])
Precision <- c(def_matrix$byClass['Precision'], opt_matrix$byClass['Precision'])
compare_df <- data.frame(Label, Accuracy, Sensitivity, Specificity, Precision)
compare_df
```

```
##            Label  Accuracy Sensitivity Specificity Precision
## 1 Default cutoff 0.7084271   0.7463876   0.6701807 0.6951276
## 2 Optimal cutoff 0.7084271   0.7463876   0.6701807 0.6951276
```

```
lr.accuracy <- opt_matrix$overall['Accuracy']
lr.sensitivity <- opt_matrix$byClass['Sensitivity']
lr.specificity <- opt_matrix$byClass['Specificity']
```

With the optimal cutoff, the majority of evaluation values are improved and they seem to prove that this Logistic Regression model works well on our data. Interestingly, the specificity has risen from 0.665 to 0.725 which is the biggest change among others.

```
acc.perf = ROCR::performance(pr, measure = "acc")
plot(acc.perf)
abline(v=c(0.5, optcut), col=c("blue", "red"))
legend(0, 0.7, legend = c("cut=0.5","optimal cut"), lty=1:1, cex=0.8, col=c("blue","red"))
```




# 5.4 Boosted Tree Model

The ideal of the boosted tree model is to build trees sequentially. Given a current model, a new tree will be fitted to the residuals firstly and then be added to the model to update the residuals. This model does not require feature scaling and is also robust to outiler. The more important thing is that the boosted decision tree is usually more accurate than the normal decision tree.

# parameter tuning

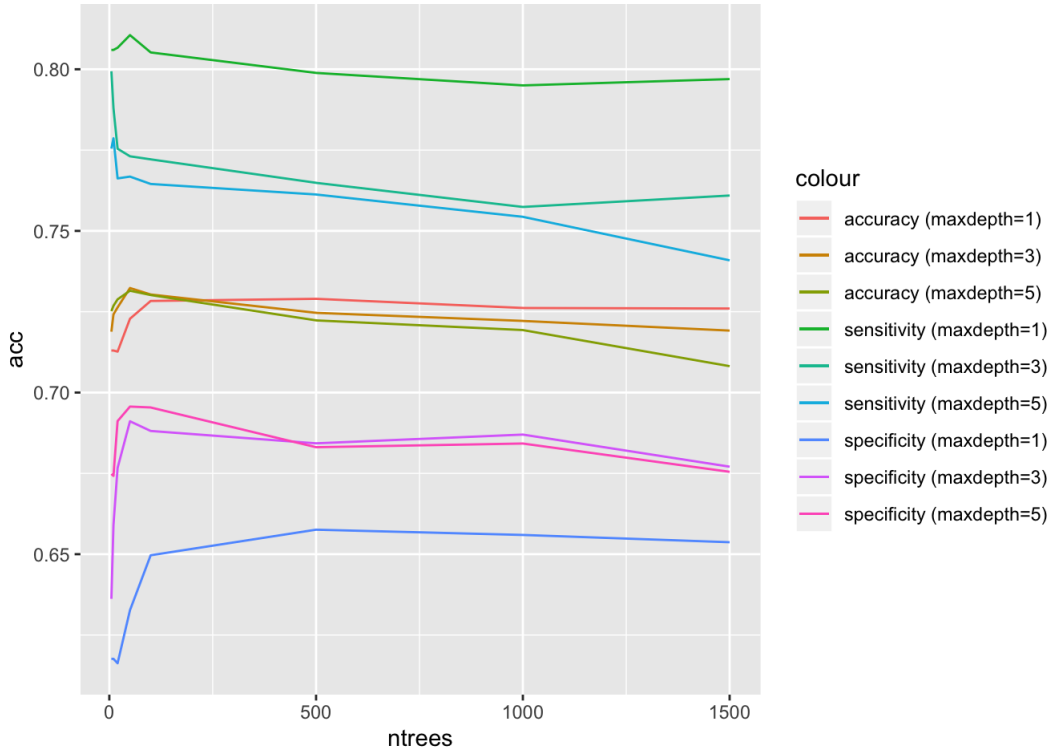
```
library(gbm)
```

```
## Loaded gbm 2.1.5
```

```
data_gbm = data_gbm_0
data_gbm <- data_gbm[!(data_gbm$ap_hi>370) & !(data_gbm$ap_hi<0) ,]
data_gbm <- data_gbm[!(data_gbm$ap_lo>360) & !(data_gbm$ap_hi< -2) ,]
data_gbm <- data_gbm[,-c(1)]
data_gbm$age<-data_gbm$age/365
data_gbm$age<-round(data_gbm$age,digits = 0)

subsample_gbm <- data_gbm[sample(nrow(data_gbm),10000),]

inTrain <- createDataPartition(subsample_gbm$cardio, p=0.6)[[1]]
trainData_gbm <- subsample_gbm[inTrain,]
testData_gbm <-  subsample_gbm[-inTrain,]
```

```
set.seed(123)
fold = createFolds(trainData_gbm$cardio, k=10)
acc_ntrees.boost = array(-16:-1,dim=c(3,8))
sensitivity_ntrees.boost = array(-16:-1,dim=c(3,8))
specificity_ntrees.boost = array(-16:-1,dim=c(3,8))
ntrees = c(5, 10, 20, 50, 100, 500, 1000, 1500)
depth = c(1, 3, 5)

set.seed(123)
for (depth_index in 1:length(depth)){
  for (ntrees_index in 1:length(ntrees)){
    acc.boost.cv = c()
    sensitivity.boost.cv =c()
    specificity.boost.cv =c()

    for (i in 1:length(fold)){
      set.seed(123)
      model.boost = gbm(cardio~., data = trainData_gbm[-fold[[i]],], distribution="bernoulli",
                        n.trees=ntrees[ntrees_index], interaction.depth=depth[depth_index])
      preds.boost.cv = predict(model.boost, newdata = trainData_gbm[fold[[i]],-12],
                               n.trees=ntrees[ntrees_index], type="response")

      for (j in 1:length(preds.boost.cv)) {
        if(preds.boost.cv[j] > 0.5)
          {preds.boost.cv[j] = 1}
        else
          {preds.boost.cv[j] = 0}
      }

      acc.boost.cv = c(acc.boost.cv, sum(preds.boost.cv==trainData_gbm[fold[[i]],12])/length(preds.boost.c
v))

      conf_mat = table(preds.boost.cv, trainData_gbm[fold[[i]],12])
      sensitivity = caret::sensitivity(conf_mat)
      sensitivity.boost.cv = c(sensitivity.boost.cv, sensitivity)
      specificity = caret::specificity(conf_mat)
      specificity.boost.cv = c(specificity.boost.cv, specificity)

    }

  acc_ntrees.boost[depth_index, ntrees_index] = mean(acc.boost.cv)
  sensitivity_ntrees.boost[depth_index, ntrees_index]  = mean(sensitivity.boost.cv)
  specificity_ntrees.boost[depth_index, ntrees_index]  = mean(specificity.boost.cv)

  }
}




df1 = data.frame("ntrees"=ntrees,
                 "acc_d1"=acc_ntrees.boost[1,],
                 "sen_d1"=sensitivity_ntrees.boost[1, ], "spe_d1"=specificity_ntrees.boost[1, ],

                 "acc_d3"=acc_ntrees.boost[2,],
                 "sen_d3"=sensitivity_ntrees.boost[2, ], "spe_d3"=specificity_ntrees.boost[2, ],

                 "acc_d5"=acc_ntrees.boost[3,],
                 "sen_d5"=sensitivity_ntrees.boost[3, ], "spe_d5"=specificity_ntrees.boost[3, ])
```

```
ggplot(df1, aes(ntrees, acc)) +
  geom_line(aes(y = acc_d1, colour = "accuracy (maxdepth=1)")) +
  geom_line(aes(y = sen_d1, colour = "sensitivity (maxdepth=1)")) +
  geom_line(aes(y = spe_d1, colour = "specificity (maxdepth=1)")) +

  geom_line(aes(y = acc_d3, colour = "accuracy (maxdepth=3)")) +
  geom_line(aes(y = sen_d3, colour = "sensitivity (maxdepth=3)")) +
  geom_line(aes(y = spe_d3, colour = "specificity (maxdepth=3)")) +

  geom_line(aes(y = acc_d5, colour = "accuracy (maxdepth=5)")) +
  geom_line(aes(y = sen_d5, colour = "sensitivity (maxdepth=5)")) +
  geom_line(aes(y = spe_d5, colour = "specificity (maxdepth=5)"))
```




To train the model, we firstly split the training set further into 10 folds (subsets) and undertake 10-fold cross validation process to find the best combination of the number of trees (ntrees) and the maximum depth of each tree that maximizes accuracy, sensitivity and specificity. By ploting accuracy, sensitivity and specificity against the number of tress, we obtain the line chart above.

According to the chart above, when the maximum depth is small, the model requires a large number of trees to improve the accuracy. When the maximum depth is large, the model can reaches the highest accuracy with a small number of trees, but the accruacy will decrease if the number of trees increases further.

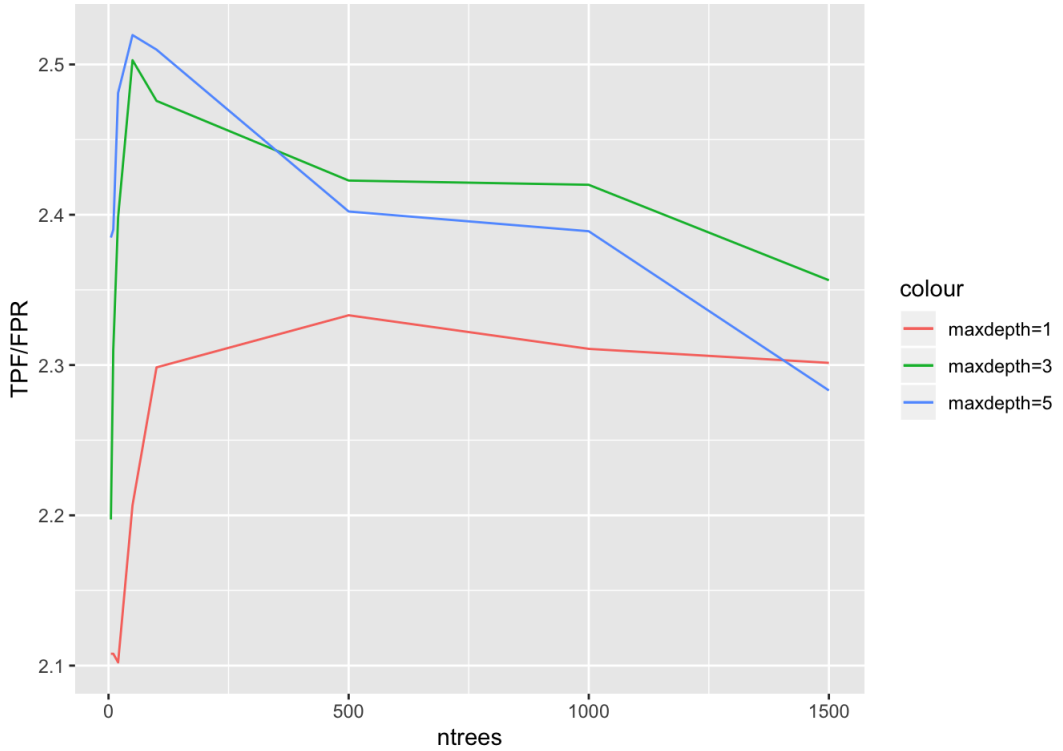
According to the chart above, a larger maximum depth will lead to a higher sensitivity, and a larger number of trees will lead to lower sensitivity. The model with (maximum depth = 1) always has higher sensitivity than the model with (maximum depth = 3 or 5) regardless the number of trees.

Becasue there is no combination of ntrees and maximum depth that maximize accuracy, sensitivity and specificity simultaneously, we introduce the ratio of TPR/FPR to identify the best combination of parameters.

```
TPR = sensitivity_ntrees.boost
FPR = 1-specificity_ntrees.boost
ROC = TPR/FPR
ROC
```

```
##          [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]
## [1,] 2.107875 2.107875 2.102159 2.206741 2.298438 2.333126 2.310771
## [2,] 2.197239 2.310728 2.398433 2.502815 2.475779 2.422770 2.419937
## [3,] 2.384824 2.390145 2.481126 2.519555 2.509864 2.402186 2.389027
##          [,8]
## [1,] 2.301437
## [2,] 2.356428
## [3,] 2.283089
```

```
df2 = data.frame("ntrees"=ntrees, "roc_d1"=ROC[1, ], "roc_d3"=ROC[2, ], "roc_d5"=ROC[3, ])

ggplot(df2, aes(ntrees, TPF/FPR)) +
  geom_line(aes(y = roc_d1, colour = "maxdepth=1")) +
  geom_line(aes(y = roc_d3, colour = "maxdepth=3")) +
  geom_line(aes(y = roc_d5, colour = "maxdepth=5"))
```




According to the graph above, when (madximum depth = 5) and (nuber of trees = 100), the model has the highest ratio of TPF/FPR. Therefore, we decided to use (madximum depth = 5) and (nuber of trees = 100) as the parameters of our boosted tree model.

# Testing

```
set.seed(123)
boosting_test = gbm(cardio~., data = trainData_gbm, distribution="bernoulli", n.trees=100, interaction.depth
=5)
preds_boosting_test = predict(model.boost, newdata = testData_gbm[,-12], n.trees=100, type="response")
for (j in 1:length(preds_boosting_test)) {
  if(preds_boosting_test[j] > 0.5)
    {preds_boosting_test[j] = 1}
  else
    {preds_boosting_test[j] = 0}
}
caret::confusionMatrix(table(preds_boosting_test, testData_gbm[,12]))
```

```
## Confusion Matrix and Statistics
##
##
## preds_boosting_test    0    1
##                    0 1568  647
##                    1  476 1309
##
##                Accuracy : 0.7192
##                  95% CI : (0.705, 0.7331)
##     No Information Rate : 0.511
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4372
##
##  Mcnemar's Test P-Value : 3.917e-07
##
##             Sensitivity : 0.7671
##             Specificity : 0.6692
##          Pos Pred Value : 0.7079
##          Neg Pred Value : 0.7333
##              Prevalence : 0.5110
##          Detection Rate : 0.3920
##    Detection Prevalence : 0.5537
##       Balanced Accuracy : 0.7182
##
##        'Positive' Class : 0
##
```

```
Boo.cm <- caret::confusionMatrix(table(preds_boosting_test, testData_gbm[,12]))

Boo.accuracy <- Boo.cm$overall['Accuracy']
Boo.sensitivity <- Boo.cm$byClass['Sensitivity']
Boo.specificity <- Boo.cm$byClass['Specificity']
```

# model evaluation - boosted tree

To evaluate the performance of the boosting model, we firstly choose (ntrees=100) and (maximum depth = 5) based on the parameter tuning result and train the model using the training set prepared before. After that, we use the testing set for prediction. The testing result is shown as above. There are 3 evaluation metrics, which are accuracy, sensitivity and specificity. The testing accuracy is 73.22%, which means 73.22% of predicitons are correct. The sensitivity is 74.34%, which means 74.34% of sick people are correctly identified as having the condition. The specificity is 72.11%, which means 72.11% of health people are correctly identified as not having the condition.

# 5.5 Support Vector Machine

Support vector machine (SVM) is a supervised machine learning method which trains the model into a non-probabilistic binary linear classifier. The basic logic behind SVM is to find a hyperplane which can maximize the interval specified in feature space. For non-linear separable problem, SVM uses Kernel Trick which transforms the feature space into a higher dimension space.

# Parameter Tuning

# Kernals:

Kernel function is used to project features space into another vector space, which is usually a higher dimensional space such that training points can be linear separable. Using appropriate kernel function can not only limit the calculation complexity, but also map the data into an appropriate dimension so they can be well classified.

After trying the four kernel functions, the RBF kernel was choosen as it achieved the highest accuracy of roughly 0.73 on training set.

```
model <- svm(cardio ~ ., data = trainData, kernel= "polynomia",
             type = "C-classification", gamma = 0.25, cost = 0.7, cross = 10)
mean(model$accuracies)
```

```
## [1] 72.12141
```

```
model <- svm(cardio ~ ., data = trainData, kernel= "radial",
             type = "C-classification", gamma = 0.25, cost = 0.7, cross = 10)
mean(model$accuracies)
```

```
## [1] 72.95502
```

```
model <- svm(cardio ~ ., data = trainData, kernel= "sigmoid",
             type = "C-classification", gamma = 0.25, cost = 0.7, cross = 10)
mean(model$accuracies)
```

```
## [1] 63.08902
```

```
model <- svm(cardio ~ ., data = trainData, kernel= "linear",
             type = "C-classification", gamma = 0.25, cost = 0.7, cross = 10)
mean(model$accuracies)
```

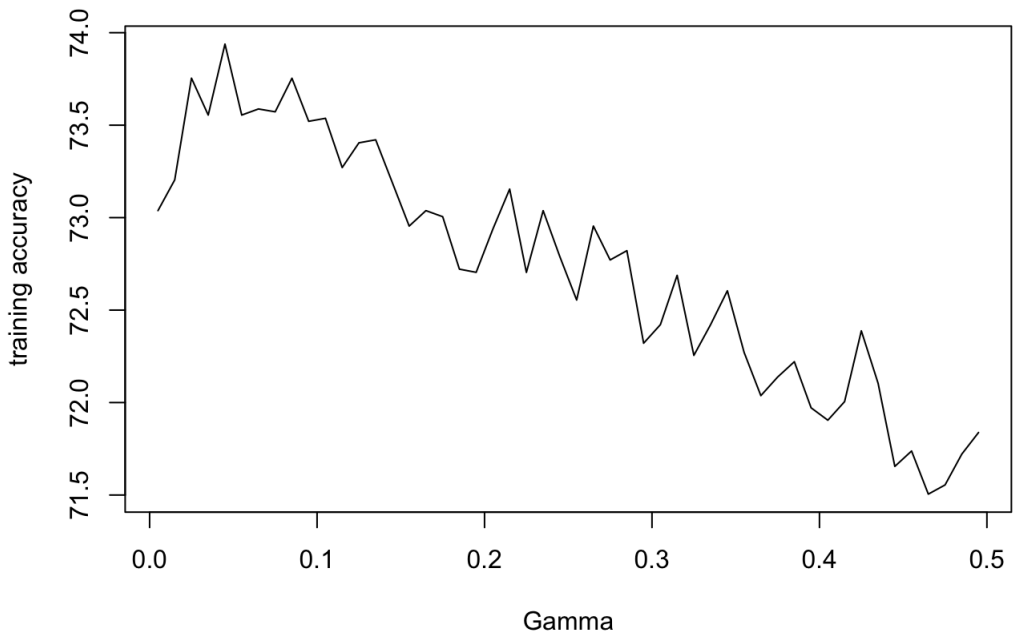
```
## [1] 72.45408
```

# Parameter tuning - gamma

Here the goal for tuning both gamma and cost is to achieve best training accurcy. As it can see in the RBF formula above, if RBM is chosen there would be a parameter Gamma. The gamma parameter defines how far the influence of a single training example reaches, with low values meaning ‘far’ and high values meaning ‘close’. We can briefly say that the larger gamma is, the less support vector the model has. The smaller gamma is, the more support vector there would be. Gamma has an influence on model efficiency and accuracy.

To find optimal Gamma, we try numbers from 0.1 to 5 with a step size 0.1. From the plot below we draw the optimal gamma. When gamma keeps growing after the optiaml point, training accuracy will see a decrease.

```
mean.acc.gamma <- c()
gamma.list <- seq(0.005, 0.5, 0.01)

for(gamma in gamma.list) {
  model <- svm(cardio ~ ., data = trainData, kernel= "radial", cross = 10,
               type = "C-classification", gamma = gamma)
  mean.acc.gamma <- c(mean.acc.gamma, mean(model$accuracies))
}
plot(seq(0.005, 0.5, 0.01), mean.acc.gamma, type = "l", xlab = "Gamma",ylab = "training accuracy")
```



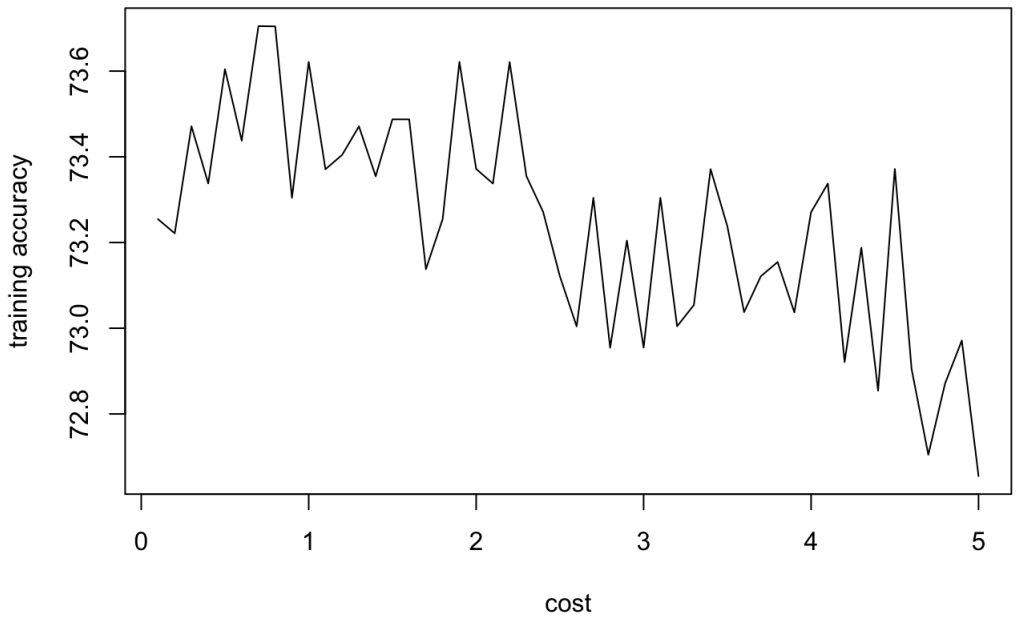
```
opt_gam <- gamma.list[which(mean.acc.gamma == max(mean.acc.gamma))]
```

# Cost

Cost is a constant of the regularization term in the Lagrange formulation. The Cost parameter behaves as a regularization parameter in the SVM. For larger values of Cost, a smaller margin will be accepted if the decision function is better at classifying all training points correctly. A lower Cost will encourage a larger margin. In other words, lower Cost will lead to a simpler decision function and maybe a lower training accuracy.

To find the optimal Cost, we try values from 0.1 to 5 with a step wise 0.1. The best Cost value fall on some point between 0 and 1 where training accuracy is roughly 0.73. We than use the optimal gamma value to fit the model next.

```
mean.acc <- c()
cost.list <- seq(0.1, 5, 0.1)

for(cost in cost.list) {
  model <- svm(cardio ~ ., data = trainData, kernel= "radial", cross = 10,
               type = "C-classification", cost = cost)
  mean.acc <- c(mean.acc, mean(model$accuracies))
}

plot(seq(0.1, 5, 0.1), mean.acc, type = "l", xlab = "cost",ylab = "training accuracy")
```




```
opt_cost <- cost.list[which(mean.acc == max(mean.acc))]
```

```
model <- svm(cardio ~ ., data = trainData, kernel= "radial",
             type = "C-classification", gamma = opt_gam, cost = opt_cost)
pred <- predict(model, testData[1:11])
sum(pred == testData[,12])/length(testData[,12])
```

```
## [1] 0.7226807
```

# Model evaluation - SVM

5 evaluation metrics were examined on testing set to evaluate the model performance: accuracy, sensitivity, specificity, precision and f1. They are as follows. We can see that the Sensitivity is highier than Specificity. The probability of False Negative is less than True Possitive. which means it is more likely to predict the no-disaster people as no-disaster, but less likely to detect the cardiovascular disaster people as disaster. General, this model is not sensitive enough to detect a protential cardiovascular disaster.

```
svm.pred <- as.factor(pred)
svm.test.y <- as.factor(testData$cardio)
def_matrix <- caret::confusionMatrix(svm.pred, svm.test.y, positive="0", mode="everything")
def_matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1563  665
##          1  444 1327
##
##                Accuracy : 0.7227
##                  95% CI : (0.7085, 0.7365)
##     No Information Rate : 0.5019
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4451
##
##  Mcnemar's Test P-Value : 3.941e-11
##
##             Sensitivity : 0.7788
##             Specificity : 0.6662
##          Pos Pred Value : 0.7015
##          Neg Pred Value : 0.7493
##               Precision : 0.7015
##                  Recall : 0.7788
##                      F1 : 0.7381
##              Prevalence : 0.5019
##          Detection Rate : 0.3908
##    Detection Prevalence : 0.5571
##       Balanced Accuracy : 0.7225
##
##        'Positive' Class : 0
##
```

```
svm.cm <- def_matrix
svm.accuracy <- svm.cm$overall['Accuracy']
svm.sensitivity <- svm.cm$byClass['Sensitivity']
svm.specificity <- svm.cm$byClass['Specificity']
```

# 6.0 Model comparison

```
result <- matrix(c(knn.accuracy, knn.sensitivity, knn.specificity,
                   lr.accuracy,lr.sensitivity,lr.specificity,
                   rf.accuracy,rf.sensitivity,rf.specificity,
                   Boo.accuracy,Boo.sensitivity,Boo.specificity,
                   svm.accuracy,svm.sensitivity,svm.specificity),
                 ncol=3,byrow=TRUE)
colnames(result) <- c("Accuracy", "Sensitivity", "Specificity")
rownames(result) <- c("Knn","LR","RF","Boosted Tree", "SVM")
result <- as.table(result)
result
```

```
##               Accuracy Sensitivity Specificity
## Knn          0.7054264   0.7548580   0.6556225
## LR           0.7084271   0.7463876   0.6701807
## RF           0.7194299   0.7638266   0.6746988
## Boosted Tree 0.7192500   0.7671233   0.6692229
## SVM          0.7226807   0.7787743   0.6661647
```

As shown in the results above, all the metrics generated by the different models are very similar, with only minimal differences. The model that produces the highest accuracy is the boosted tree, however, it cannot be regarded as a model but rather a ensemble method of decision tree. Despite that the boosted tree method produces the highest accuracy, The random forest model also produces great results in correctly predicting whether a patient has cardiovascular disease. More importantly, although accuracy is a good indicating on evaluating model performace, in a clinical scenario, we do think that the specificity score should be the core in evaluating model

performance because it measures the amount of patients which has cardiovascular disease, and are correctly classified. We want to maximise the chance of patients being correctly classified with the disease than not having the disease. Therefore, in terms of specificity score, the logistic regression model outperformed the others, hence we will be choosing logistic regression model to fit our overall dataset.

Despite the evaluation on above metrics, in terms of running time, the logistic regression model performance was again impressive. All function were able to run within a 5 minute time frame whereas other models like the random forest and SVM took comparatively longer to run. Hence as regard to practical use and efficiency, the logistic regression model is no doubt a better choice than the other models we've tried.

# 7.0 Final model fit

```
set.seed(123)
inTrain <- createDataPartition(data$cardio, p=0.6)[[1]]
final_trainData <- data[inTrain,]
final_testData <-  data[-inTrain,]

findal_model <- glm(cardio~., data = final_trainData, family = "binomial")
final_pred <- predict(findal_model, newdata = final_testData, type = "response")
final_pred_result <- ifelse(final_pred > optcut, 1, 0)
final_pred_result_fac <- as.factor(final_pred_result)
final_test_fac <- as.factor(final_testData$cardio)
final_matrix <- caret::confusionMatrix(final_pred_result_fac, final_test_fac, positive="0", mode="everythin
g")
final_matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 10823  4501
##          1  3118  9157
##
##                Accuracy : 0.7239
##                  95% CI : (0.7186, 0.7292)
##     No Information Rate : 0.5051
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4473
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.7763
##             Specificity : 0.6704
##          Pos Pred Value : 0.7063
##          Neg Pred Value : 0.7460
##               Precision : 0.7063
##                  Recall : 0.7763
##                      F1 : 0.7397
##              Prevalence : 0.5051
##          Detection Rate : 0.3922
##    Detection Prevalence : 0.5552
##       Balanced Accuracy : 0.7234
##
##        'Positive' Class : 0
##
```

```
final_matrix$byClass['Specificity']
```

```
## Specificity
##   0.6704496
```

# 8.0 Conclusion

In conclusion, in order to build an accurate and efficient classifier for detecting cardiovascular disease, we have performed exploratory data analysis by looking at correlation and interrelationship between features, removed outliers, performed feature engineering by normalising the data, subsampled the data to facilitate model building, selected and dropped features to build models where appropriate,

built and tuned 5 different models to improve their performances, and compared across all models in terms of their accuracy, sensitivity, specificity and run time.

The final model we will use to detect cardiovascular disease is the logistic regression model with cut off point at 0.456. This is based on the consideration that all models essentially have similar performance in terms of performance metrics, but logistic regression model is efficient due to comparatively shorter run time.

Lastly, wiht the aim of improving the logistic regression model further, in the near future we will perform more parameter tuning, and test the model with more up-to-day data down the track.

# 9.0 References

1] National Heart Foundation of Australia, “Is my blood pressure normal” https://www.heartfoundation.org.au/your-heart/know-your-risks/blood-pressure/is-my-blood-pressure-normal (https://www.heartfoundation.org.au/your-heart/know-your-risks/blood-pressure/is-my-blood-pressure-normal), 2019, viewed 20 Oct 2019 [2] Narloch JA and Brandstater ME, 1995, ‘Influence of breathing technique on arterial blood pressure during heavy weight lifting’, Arch Phys Med Rehabil, vol. 76(5), pp. 457-62, viewed 19 Oct 2019, DOI: 10.1016/s0003-9993(95)80578-8 [3] Sabbah HN, Anbe DT and Stein PD, 1981, ‘Can the human right ventricle create a negative diastolic pressure suggestive of suction?’, Cathet Cardiovasc Diagn, vol. 7(3), pp. 259-67 viewed 19 Oct 2019, DOI: 10.1002/ccd.1810070305