R Notebook

Data Preprocessing.

Libraries:

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyverse)
```

```
## -- Attaching packages
---------------------------------------------------------------------------------------- tid
yverse 1.2.1 --
```

```
## v ggplot2 3.1.0     v readr   1.3.1
## v tibble  2.1.1     v purrr   0.3.2
## v tidyr   0.8.3     v stringr 1.4.0
## v ggplot2 3.1.0     v forcats 0.4.0
```

```
## -- Conflicts
-------------------------------------------------------------------------------------------
tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(ggplot2)
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
library(PerformanceAnalytics)
```

```
## Loading required package: xts
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
##
## Attaching package: 'xts'
```

```
## The following objects are masked from 'package:dplyr':
##
##     first, last
```

```
##
## Attaching package: 'PerformanceAnalytics'
```

```
## The following object is masked from 'package:graphics':
##
##     legend
```

Import dataset:

```
cardio <- read.csv("cardio_train.csv", sep = ";" )
```

Let's look at our dataset for exploration

```
head(cardio)
```

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

6 rows | 1-10 of 14 columns

```
summary(cardio)
```

```
##        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
```

Analysis of our variables :

Going by the information from head() and summary() we can see that all of our variables have numerical values but given by the description, variables like cholesterol and gluc are categorical, and gender, smoke, alco, active, cardio are binary in nature.

Here age is counted in number of days. Gender has two values, 1 and 2. 1 - women and 2 - men. Height is in cms. weight is in kgs. ap_hi and ap_low are systolic and diastolic blood pressure values. Cholestrol has 3 values: 1 : normal 2 : above normal 3 : well above normal gluc stands for level of glucose: 1: normal 2: above normal 3: well above normal Smoke is binary, if a person smokes value is 1. Similarly alco stands for alcohol consumption, and active stands for physical activity, they are both binary in nature. Cardio is the target variable that tells us whether the person has cardiovascular disease or not. Value is 1 if the person has cvd.

Looking for missing values:

```
any(is.na(cardio))
```

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

Our dataset has no missing values.

Looking for inaccuracies / inconsistency throughout the dataset:

Let's create a duplicate dataset:

```
cardio_Dup <- cardio
```

Firstly, diastolic blood pressure cannot be higher than systolic pressure.

```
cardio_Dup <- subset(cardio_Dup, ap_hi > ap_lo)
```

Also, the dataset cannot have negative or 0 measurements for blood pressure. Since 0 means no blood is being pumped.

```
cardio_Dup <- subset(cardio_Dup, ap_hi > 0 & ap_lo >0)
```

We cannot have systolic pressure more than 300

```
cardio_Dup <- subset(cardio_Dup, ap_hi < 300)
```

We cannot decide on the lower values of blood pressure, since we lack domain knowledge.

Let's find the minimum age in our dataset: We divide our variable by 365 since our age here is given in number of days.

```
cardio_Dup$age <- cardio_Dup$age/365
summary(cardio_Dup$age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   29.58   48.38   53.98   53.33   58.42   64.97
```

Our minimum age recorded here is 29.58 years old.

We remove any weight recorded less than 30 kgs.

```
cardio_Dup <- subset(cardio_Dup, weight > 30)
```

```
summary(cardio_Dup$weight)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   31.00   65.00   72.00   74.12   82.00  200.00
```

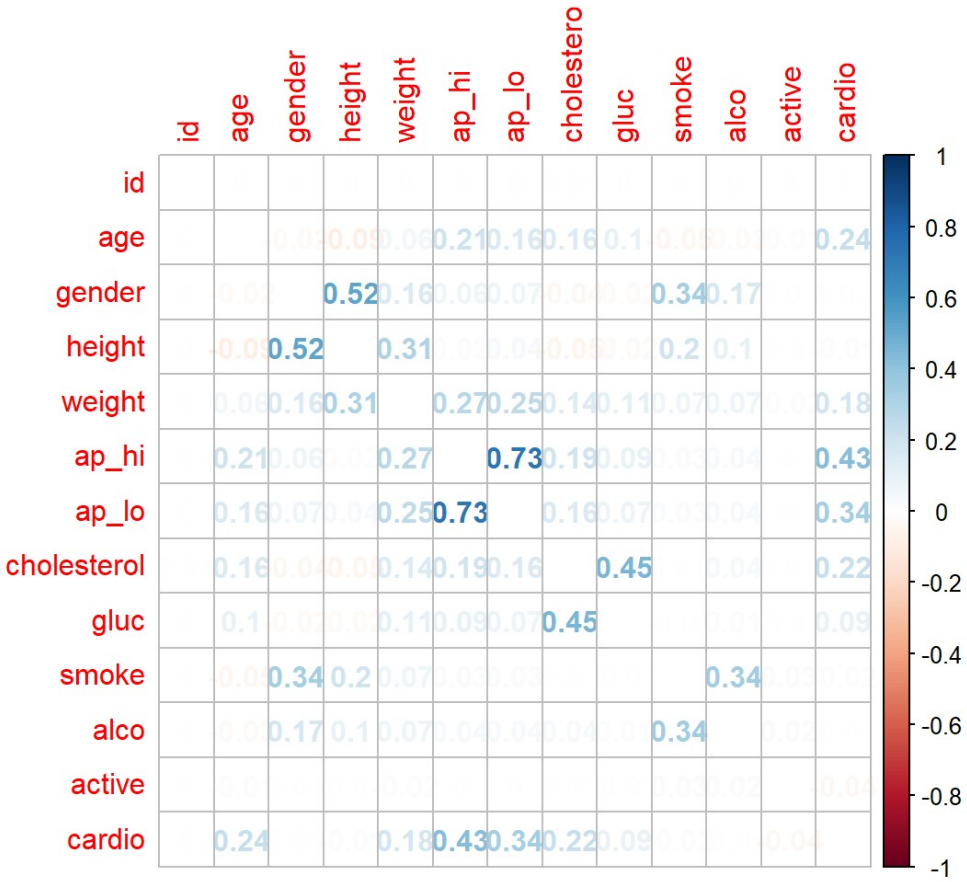
For height we remove any height less than 100 cms

```
cardio_Dup <- subset(cardio_Dup, height > 120)
```

Hence we are done cleaning and removing inconsistencies from our dataset.

Multivariate Analysis:

```
cormat <- cor(cardio_Dup[,])
corrplot(cormat, diag = FALSE, method = "number")
```



By looking at the above correlation matrix, we can see that systolic blood pressure and diastolic blood pressure is weakley correlated with our target variable cardio. Cholesterol and age also have some impact on the target variable. Cholesterol and glucose levels as have a moderate correlation.

We also create a new variable in our duplicated dataset called Body Mass Index. BMI helps us understand if a person is overweight or underweight.

```
cardio_Dup$bmi <- cardio_Dup$weight/((cardio_Dup$height/100)^2)
```

Let's see correlation between bmi and cardio

```
cor(cardio_Dup$bmi, cardio_Dup$cardio)
```

```
## [1] 0.1910442
```

Visualizations:

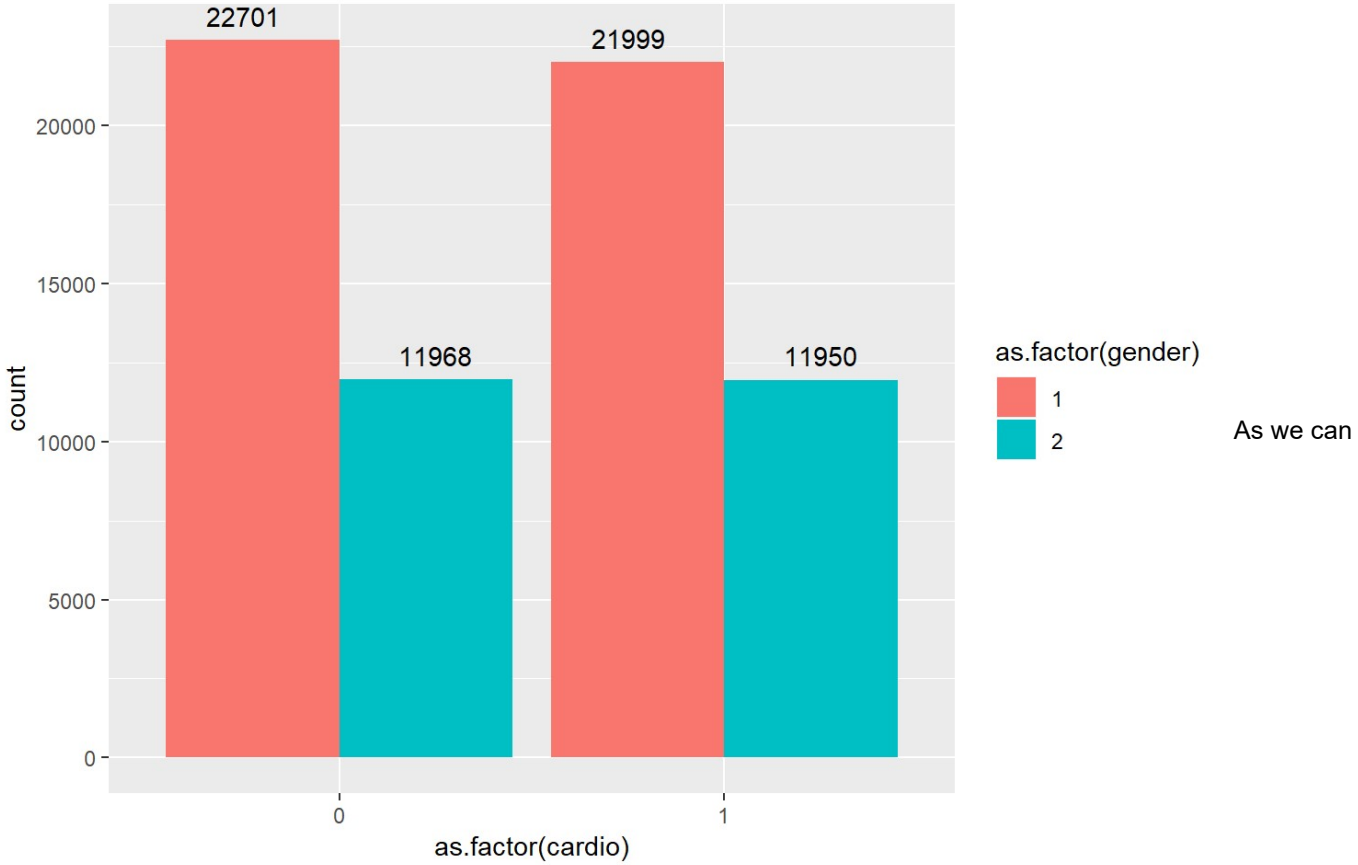
"_____"

Here, Cardio Positive cases are taken into accounts while analysis.

"_____"

barplot of cardio to see how many patients have cvd, group by their gender. with count mentioned on the bars

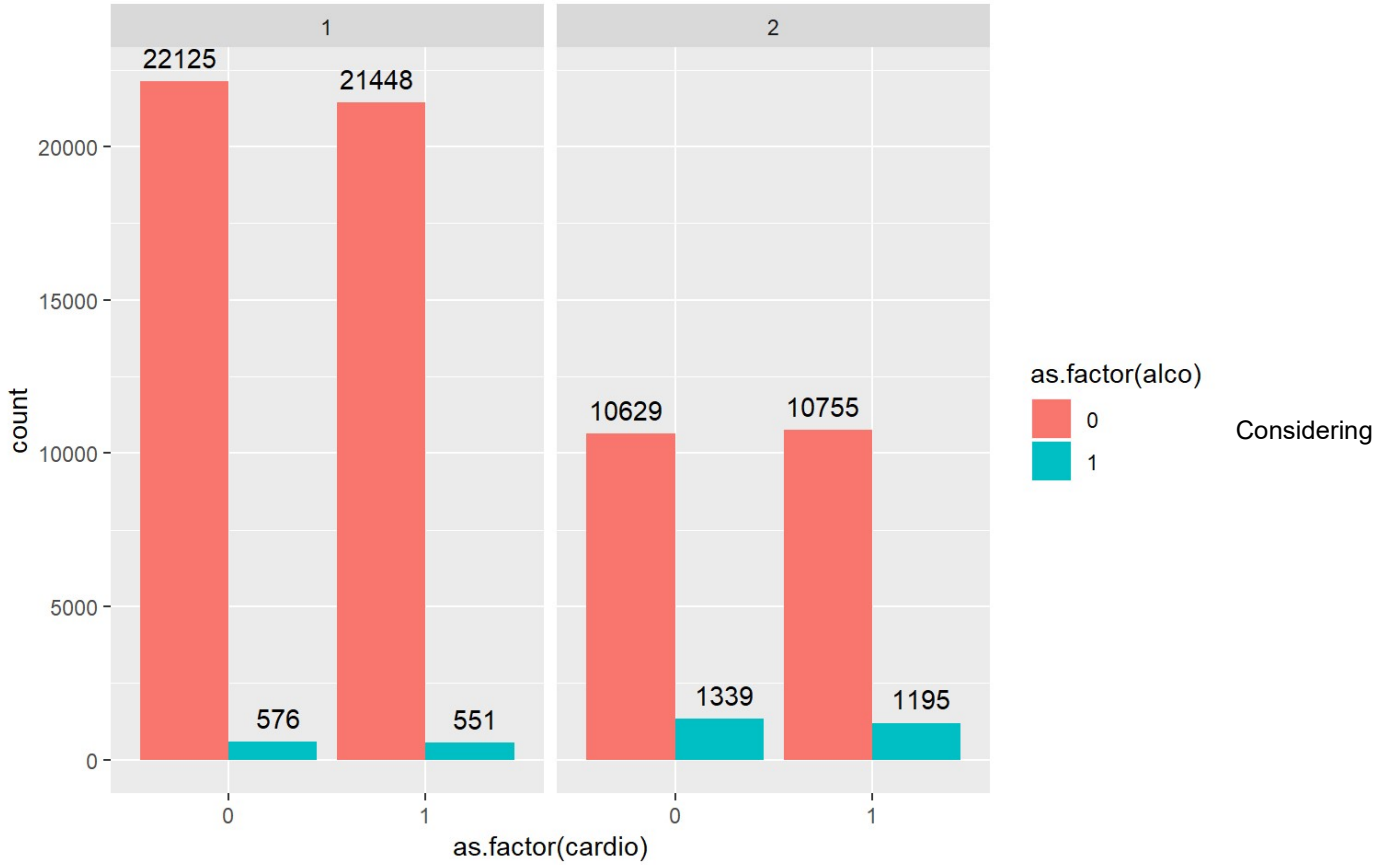
```
ggplot(cardio_Dup, aes(x = as.factor(cardio), fill = as.factor(gender))) + geom_bar(stat = "co
unt", position = position_dodge()) + geom_text(aes(label = ..count..), stat = "count", positio
n = position_dodge(width = 1), vjust = -0.7)
```



As we can see, our target variable is almost equally distributed across the genders.

"barplot of cardio wrt alcohol grouped by gender"

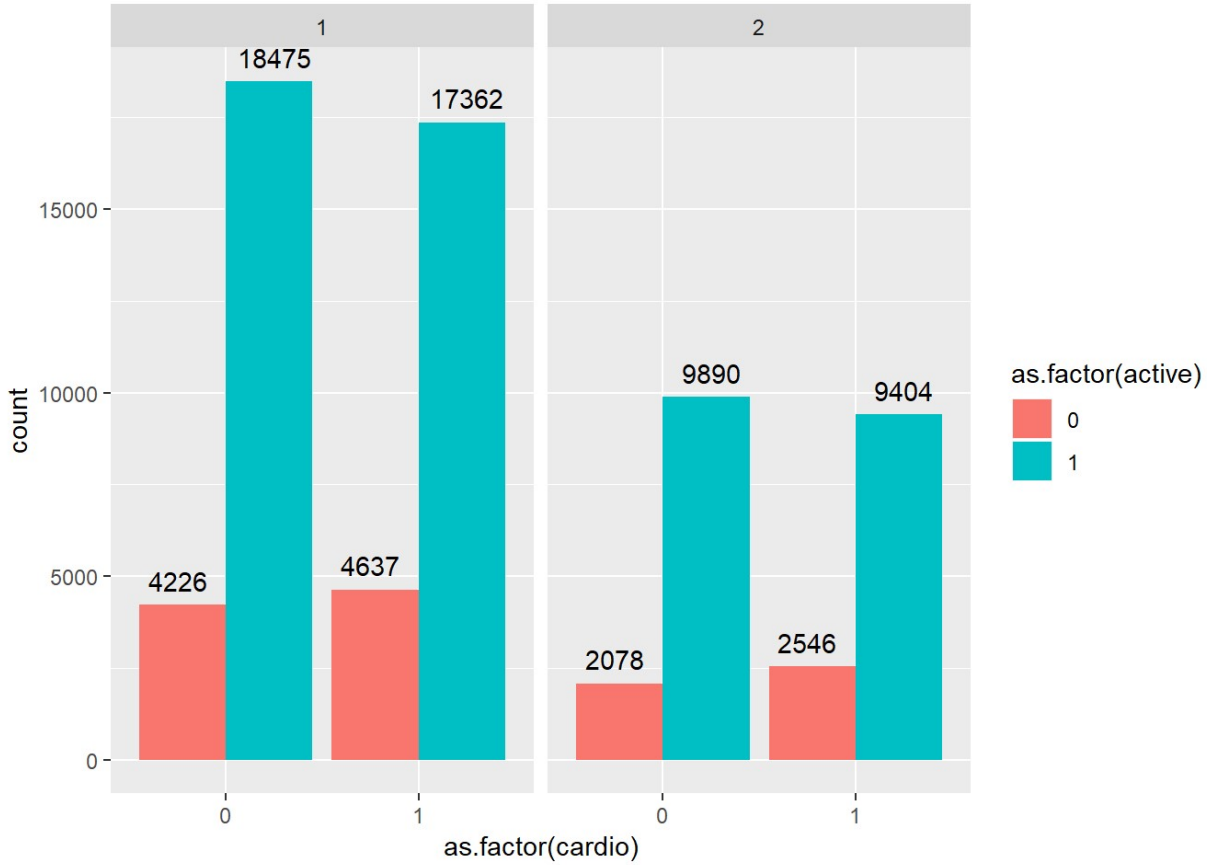
```
ggplot(cardio_Dup, aes(x = as.factor(cardio), fill = as.factor(alco))) + geom_bar(stat = "coun
t", position = position_dodge()) + geom_text(aes(label = ..count..), stat = "count", position
= position_dodge(width = 1), vjust = -0.7) + facet_grid(.~gender)
```



the female and male population seperately. No analysis can be made here, if cardiovascular diseases depend on alcohol.

“barplot of cardio wrt active grouped by gender”

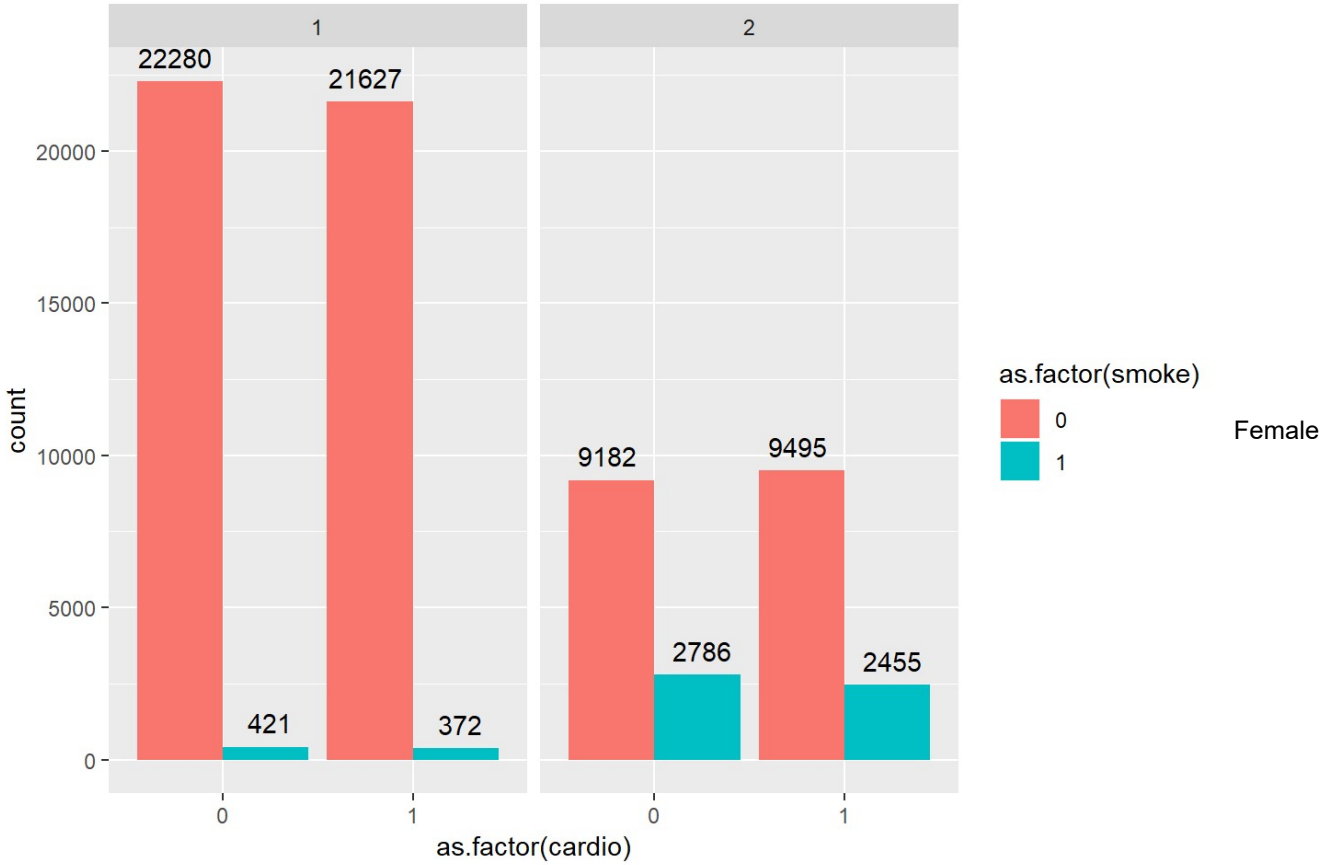
```
ggplot(cardio_Dup, aes(x = as.factor(cardio), fill = as.factor(active))) + geom_bar(stat = "co
unt", position = position_dodge()) + geom_text(aes(label = ..count..), stat = "count", positio
n = position_dodge(width = 1), vjust = -0.7) + facet_grid(.~gender)
```



In both the population, people who are actively excercising are seen to have cardio diseases. No assumptions can be made here regarding positive effects of exercising on cardiovascular diseases.

“barplot of cardio wrt smoke grouped by gender”

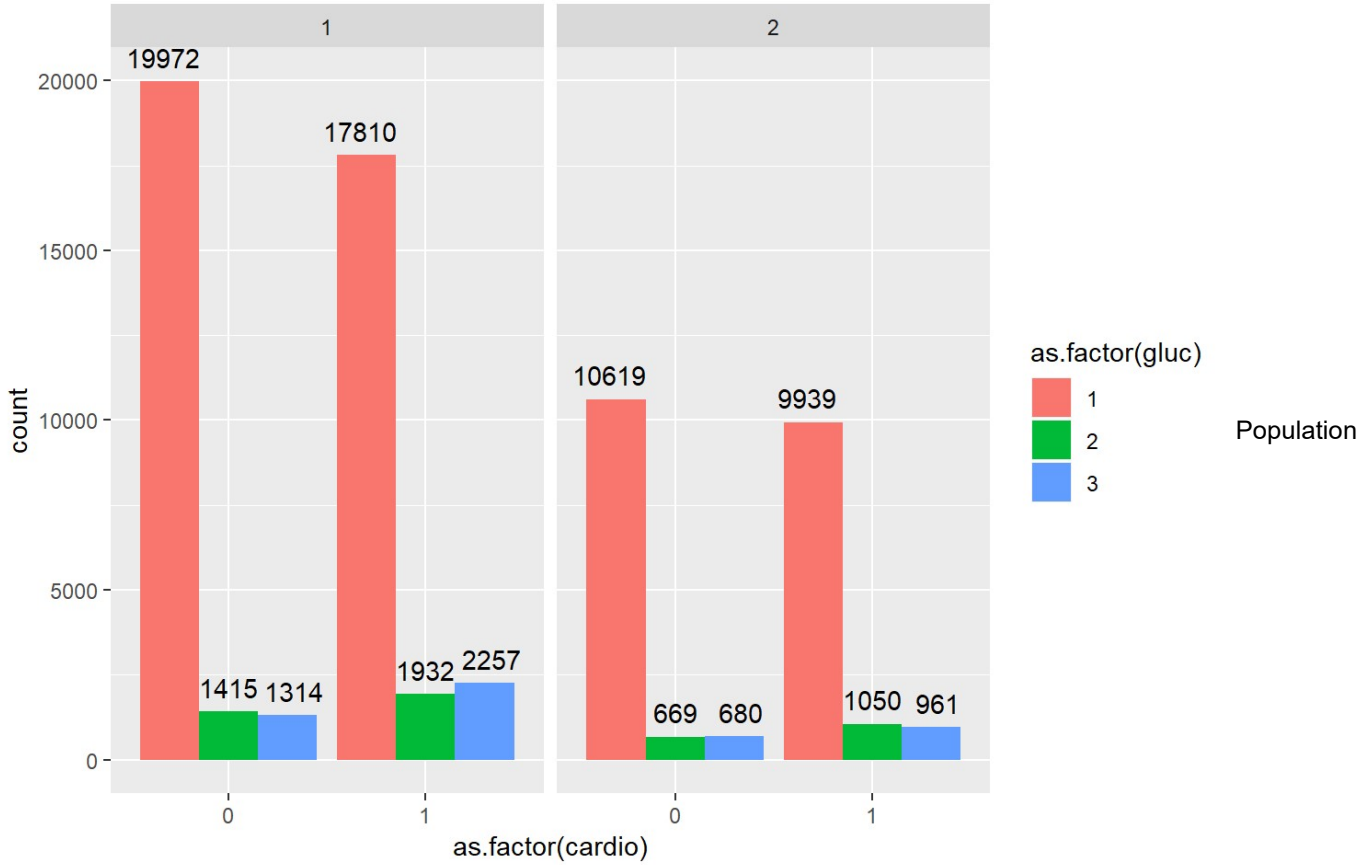
```
ggplot(cardio_Dup, aes(x = as.factor(cardio), fill = as.factor(smoke))) + geom_bar(stat = "cou
nt", position = position_dodge()) + geom_text(aes(label = ..count..), stat = "count", position
= position_dodge(width = 1), vjust = -0.7) + facet_grid(.~gender)
```



population is seem to smoke more than male population in this dataset. But the ratio of cardio positive and smokers is high in male population. Though female smoker population is more but male population shows higher cardio diseases.

“barplot of cardio wrt glucose grouped by gender”

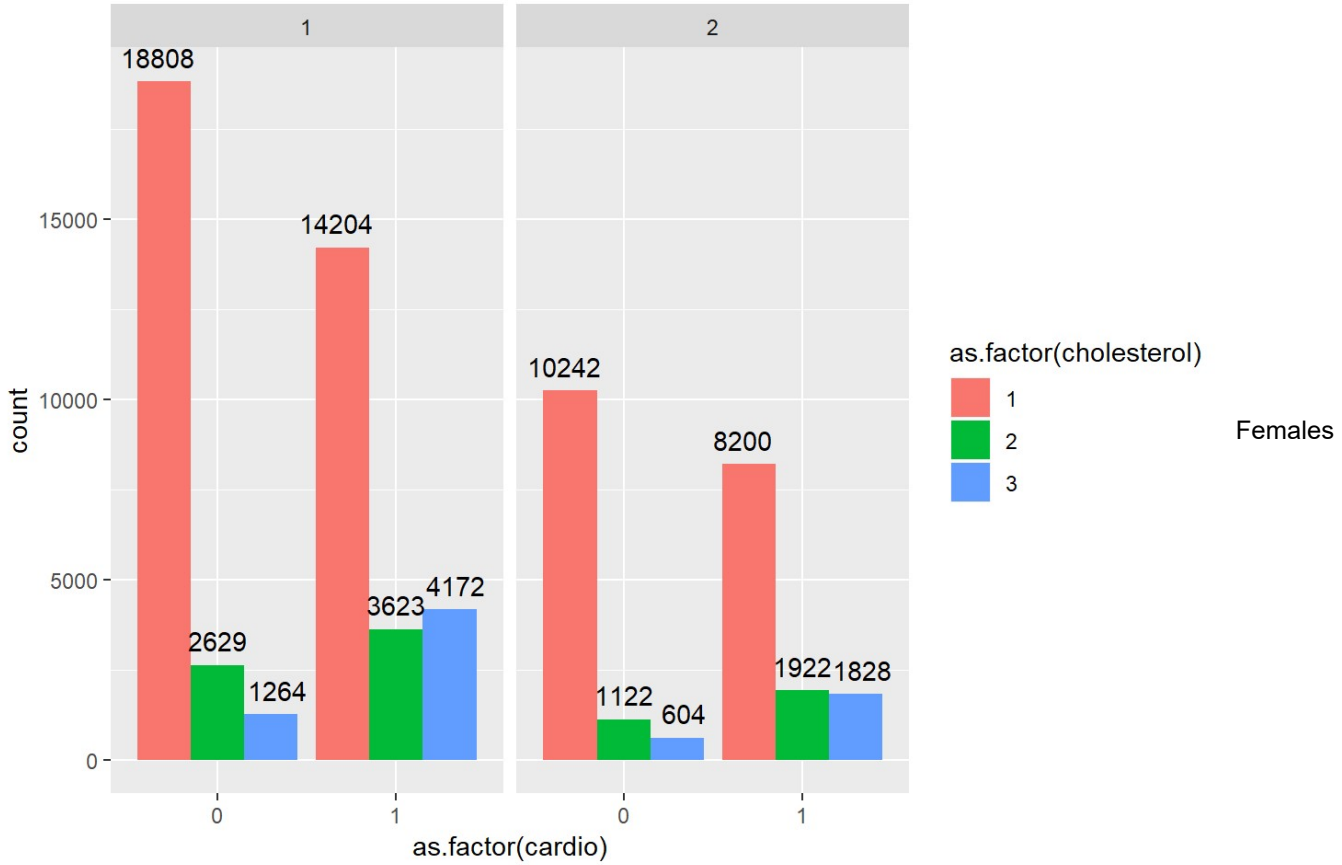
```
ggplot(cardio_Dup, aes(x = as.factor(cardio), fill = as.factor(gluc))) + geom_bar(stat = "coun
t", position = position_dodge()) + geom_text(aes(label = ..count..), stat = "count", position
= position_dodge(width = 1), vjust = -0.7) + facet_grid(.~gender)
```



with normal glucose level shows higher cases of cardio diseases when compared to the group with high and very high glucose levels.

“barplot of cardio wrt cholestrol grouped by gender”

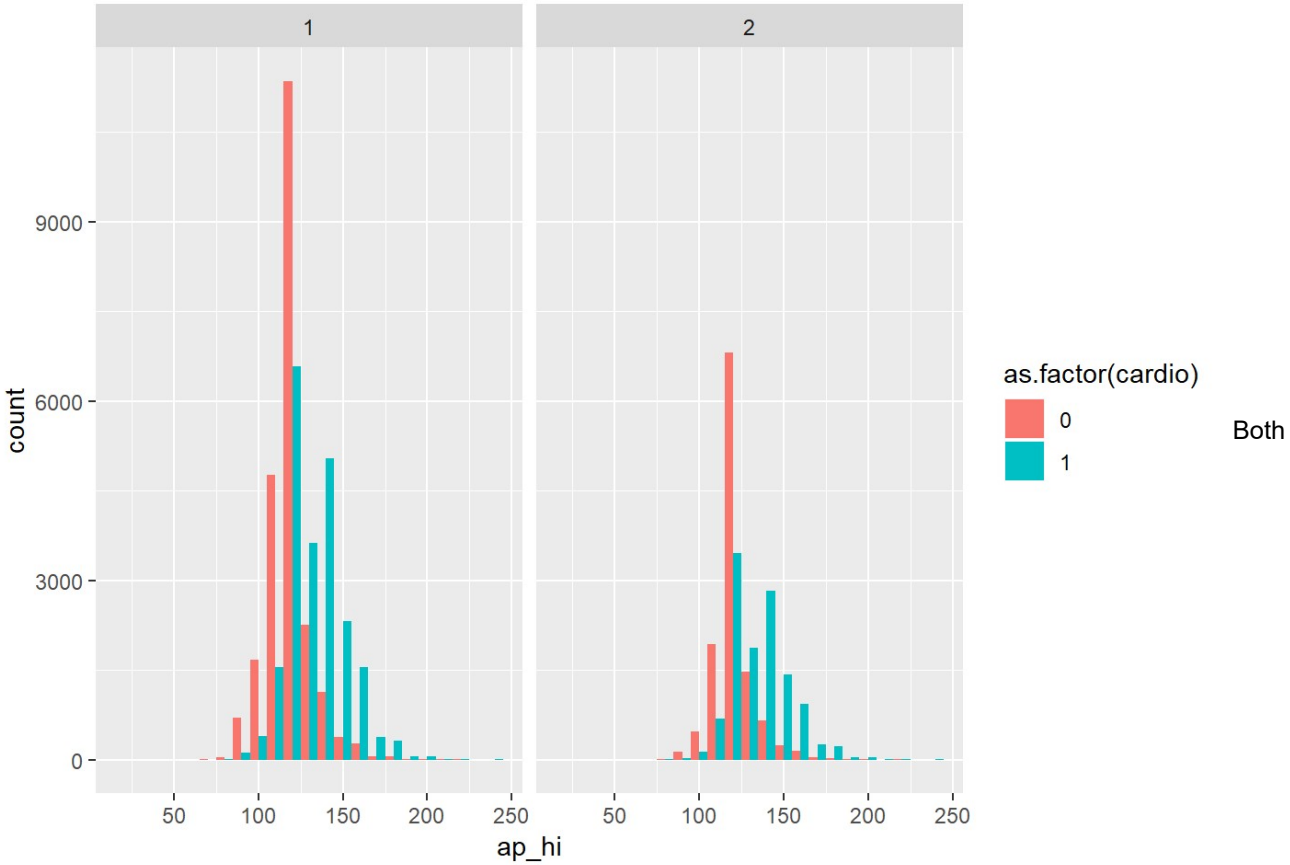
```
ggplot(cardio_Dup, aes(x = as.factor(cardio), fill = as.factor(cholesterol))) + geom_bar(stat
= "count", position = position_dodge()) + geom_text(aes(label = ..count..), stat = "count", po
sition = position_dodge(width = 1), vjust = -0.7) + facet_grid(.~gender)
```



tend to show higher cholestrol levels than males. Also, the ratio of high cholestrol and cardio diseases count is higher in female population. Looking into cardio positive class, people with normal cholestrol too have cardio diseases in abundance when compared to those with high and very high cholestrol levels.

“barplot of cardio wrt high blood pressure grouped by gender”

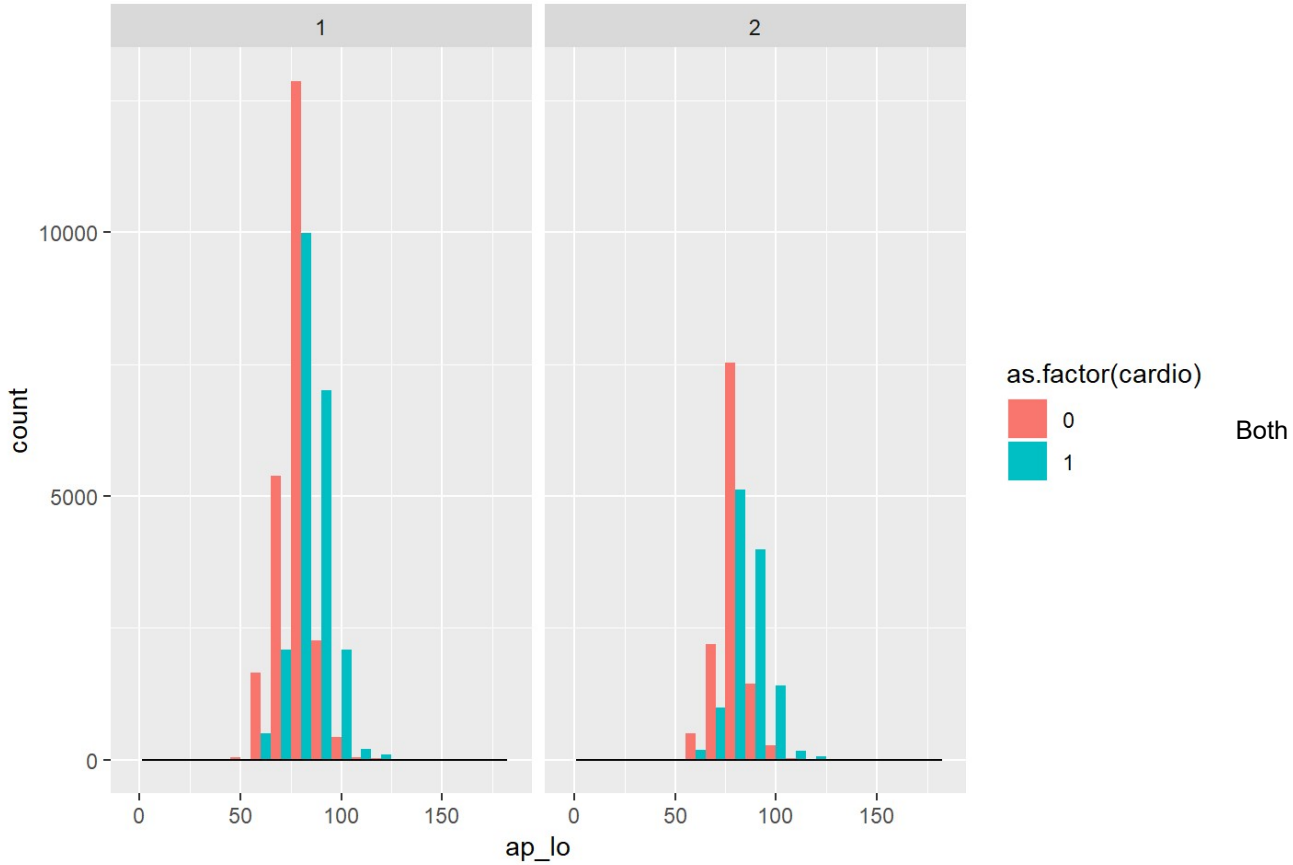
```
ggplot(cardio_Dup, aes(x = ap_hi)) + geom_histogram(binwidth = 10, aes(fill = as.factor(cardi
o)), position = "dodge") + facet_grid(.~gender)
```



genders shows right skewness in the data. People with high blood pressures are shown to have cardio diseases.

“barplot of cardio wrt low blood pressure grouped by gender”

```
ggplot(cardio_Dup, aes(x = ap_lo)) + geom_histogram(binwidth = 10, aes(fill = as.factor(cardi
o)), position = "dodge") + facet_grid(.~gender) + geom_density(alpha=0.3, fill= "black")
```

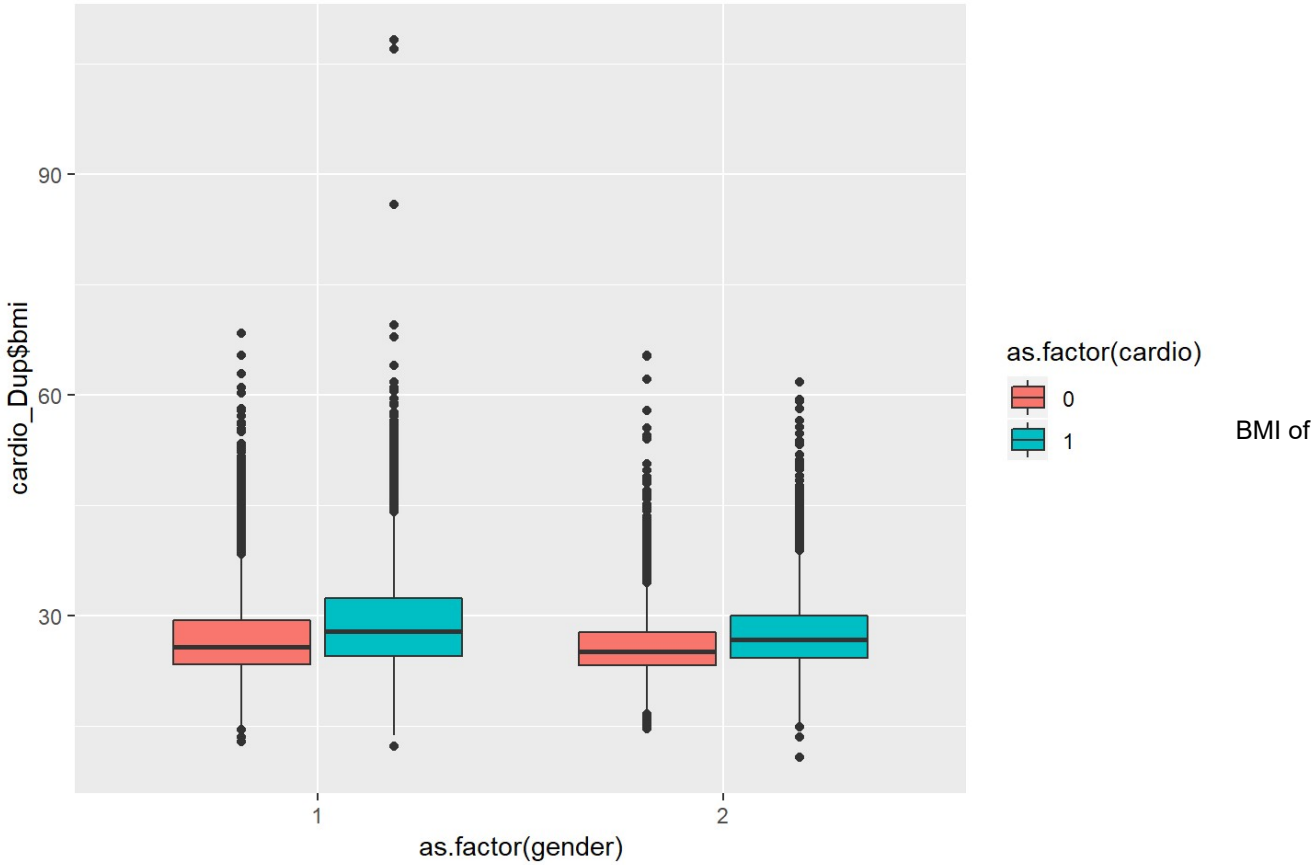


genders show a left skewness in the distribution. People with low blood pressure are shown to have cardio diseases.

"boxplot of BMI wrt gender grouped by cardio"

```
ggplot(cardio_Dup, aes(x=as.factor(gender), y=cardio_Dup$bmi)) +
    geom_boxplot(aes(fill=as.factor(cardio),position= "dodge"))
```

```
## Warning: Ignoring unknown aesthetics: position
```

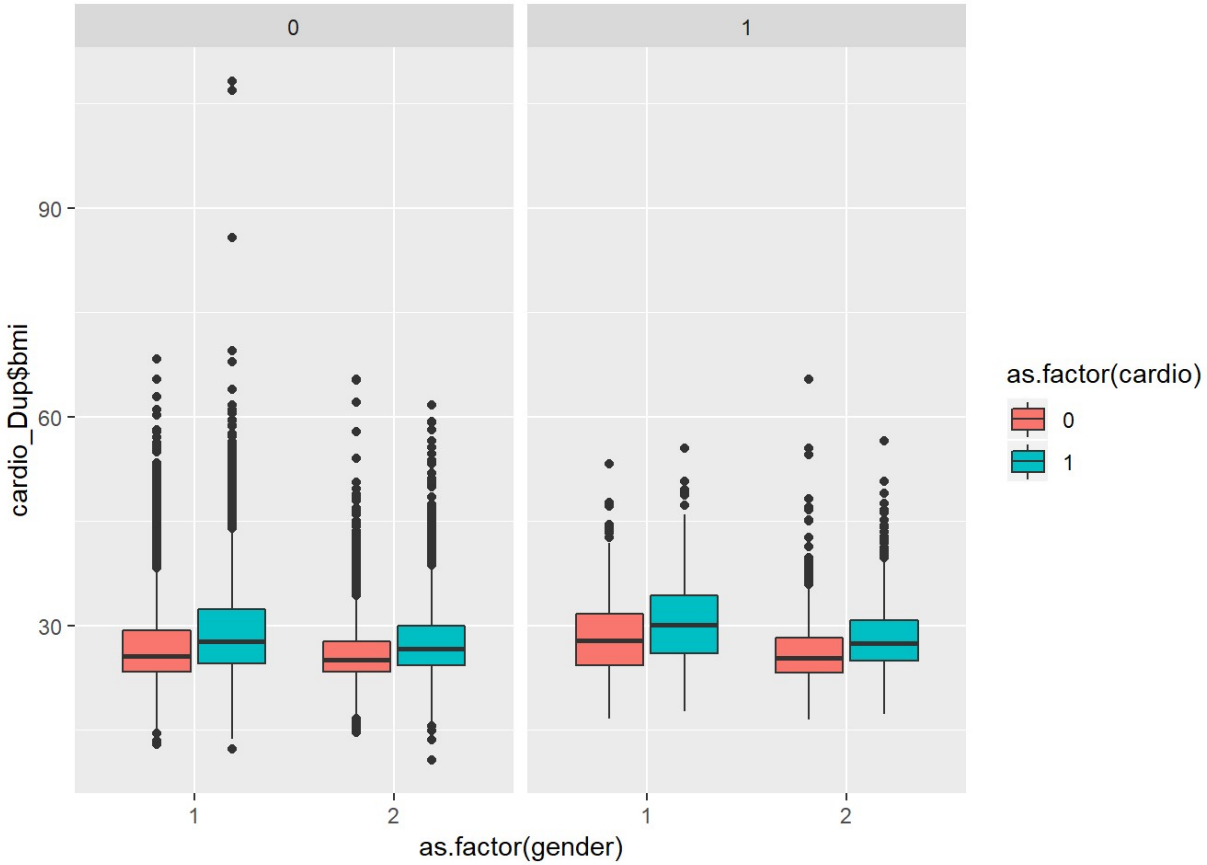


people with cardio positive is higher than that of cardio negative people. Population with higher BMI has higher chances of cardio diseases. This can be verified in the different cases shown below.

“boxplot of BMI wrt gender grouped by cardio facetgrid by alcohol”

```
ggplot(cardio_Dup, aes(x=as.factor(gender), y=cardio_Dup$bmi)) +
    geom_boxplot(aes(fill=as.factor(cardio),position= "dodge"))+ facet_wrap(~alco)
```

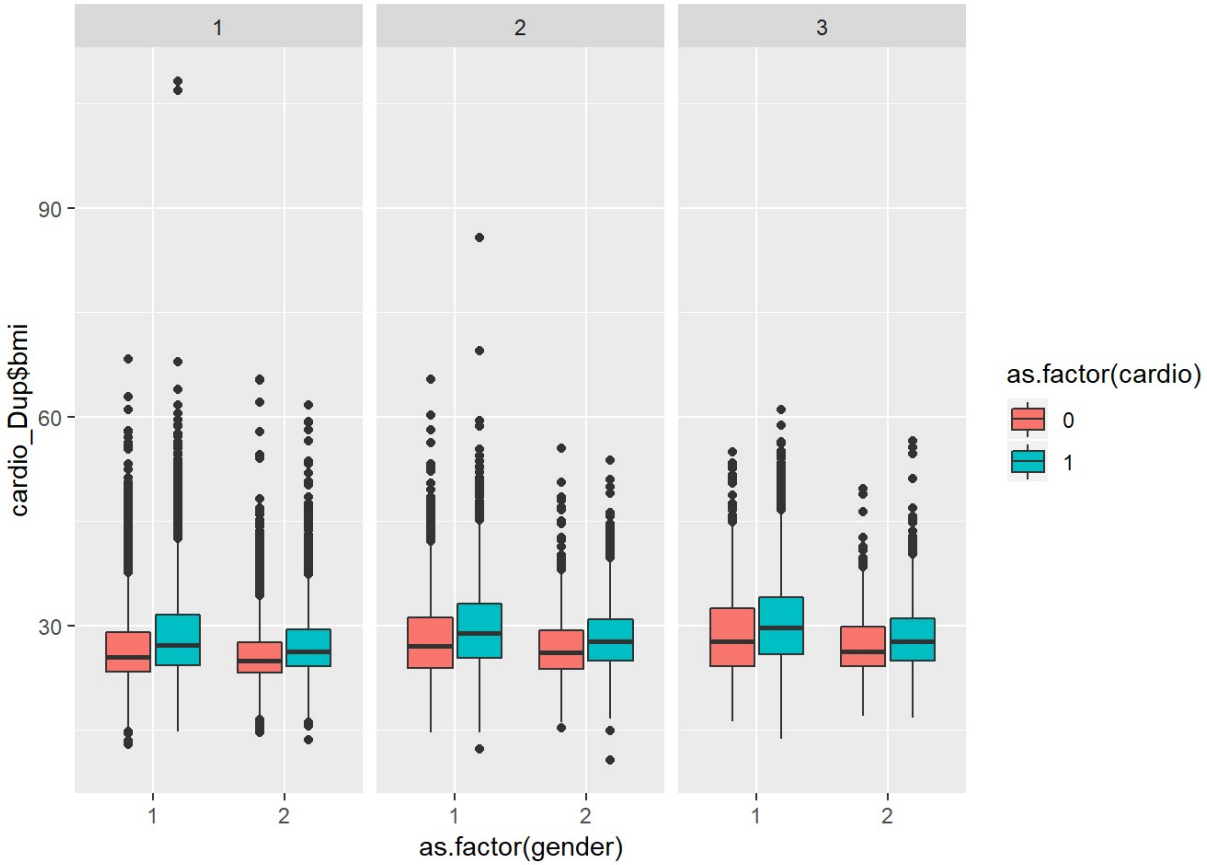
```
## Warning: Ignoring unknown aesthetics: position
```



"boxplot of BMI wrt gender grouped by cardio facetgrid by cholesterol"

```
ggplot(cardio_Dup, aes(x=as.factor(gender), y=cardio_Dup$bmi)) +
    geom_boxplot(aes(fill=as.factor(cardio),position= "dodge"))+ facet_wrap(~cholesterol)
```

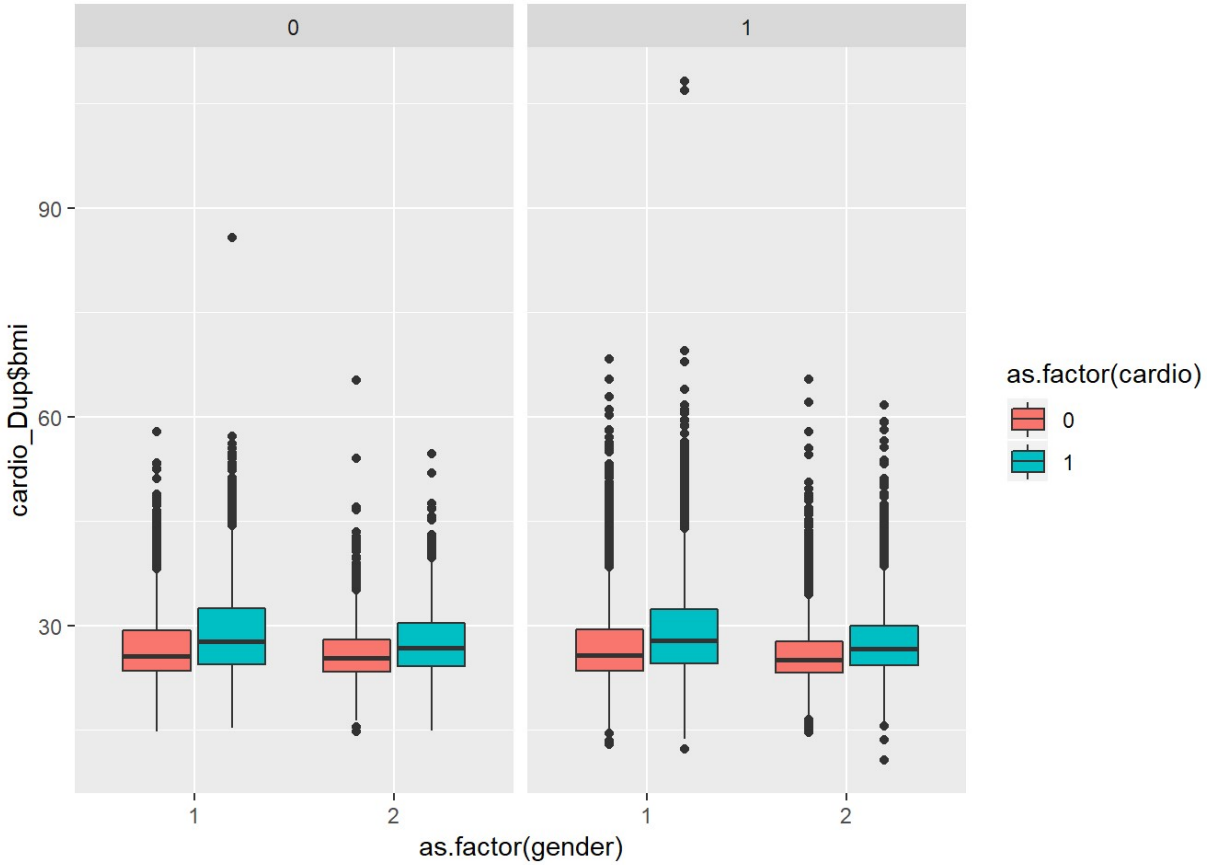
```
## Warning: Ignoring unknown aesthetics: position
```



"boxplot of BMI wrt gender grouped by cardio facetgrid by active"

```
ggplot(cardio_Dup, aes(x=as.factor(gender), y=cardio_Dup$bmi)) +
    geom_boxplot(aes(fill=as.factor(cardio),position= "dodge"))+ facet_wrap(~active)
```

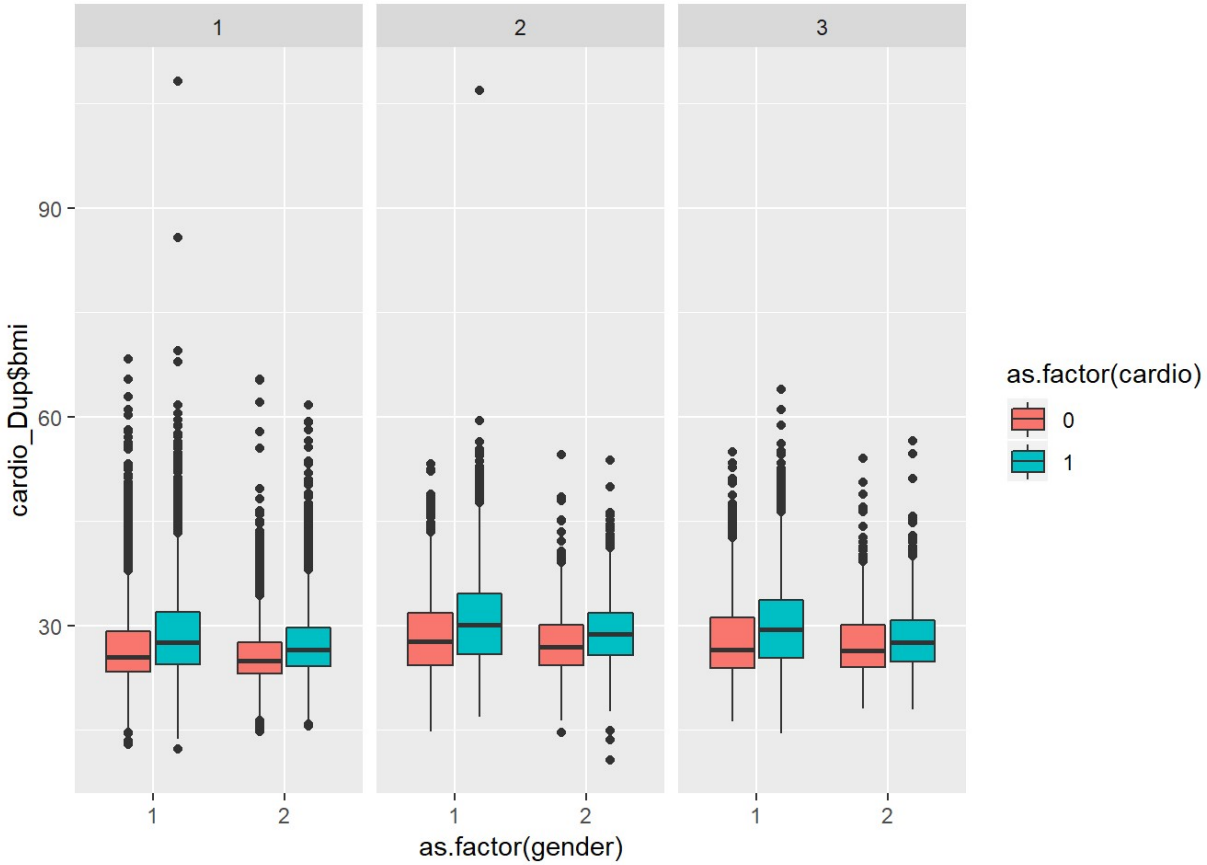
```
## Warning: Ignoring unknown aesthetics: position
```



"boxplot of BMI wrt gender grouped by cardio facetgrid by glucose"

```
ggplot(cardio_Dup, aes(x=as.factor(gender), y=cardio_Dup$bmi)) +
    geom_boxplot(aes(fill=as.factor(cardio),position= "dodge"))+ facet_wrap(~gluc)
```

```
## Warning: Ignoring unknown aesthetics: position
```



People whether or not consuming alcohol, with higher BMI are more in number than with lower BMI with cardio diseases. In all the groups of cholesterol level, people with higher BMI are more in number than with lower BMI with cardio diseases.

CLASSIFICATION TREE

```
cardio_train <- read.csv("Train_Set.csv")
cardio_valid <- read.csv("Test_Set.csv")


for( i in 1:nrow(cardio_valid)){
  if(cardio_valid$gender[i]== 1)cardio_valid$gender[i] = "F"
  if(cardio_valid$gender[i]== 2)cardio_valid$gender[i] = "M"
}
cardio_valid$gender <- as.factor(cardio_valid$gender)
cardio_valid$cholesterol <- factor(cardio_valid$cholesterol, ordered = TRUE)
cardio_valid$gluc <- factor(cardio_valid$gluc, ordered = TRUE)
cardio_valid$smoke <- as.factor(cardio_valid$smoke)
cardio_valid$alco <- as.factor(cardio_valid$alco)
cardio_valid$active <- as.factor(cardio_valid$active)
cardio_valid$cardio <- as.factor(cardio_valid$cardio)

for( i in 1:nrow(cardio_train)){
  if(cardio_train$gender[i]== 1)cardio_train$gender[i] = "F"
  if(cardio_train$gender[i]== 2)cardio_train$gender[i] = "M"
}
cardio_train$gender <- as.factor(cardio_train$gender)
cardio_train$cholesterol <- factor(cardio_train$cholesterol, ordered = TRUE)
cardio_train$gluc <- factor(cardio_train$gluc, ordered = TRUE)
cardio_train$smoke <- as.factor(cardio_train$smoke)
cardio_train$alco <- as.factor(cardio_train$alco)
cardio_train$active <- as.factor(cardio_train$active)
cardio_train$cardio <- as.factor(cardio_train$cardio)

cardio_train$X <- NULL
cardio_valid$X <- NULL
```

Creating basic Classification Tree on our dataset

```
library(rpart)
library(rpart.plot)
library(caret)
```

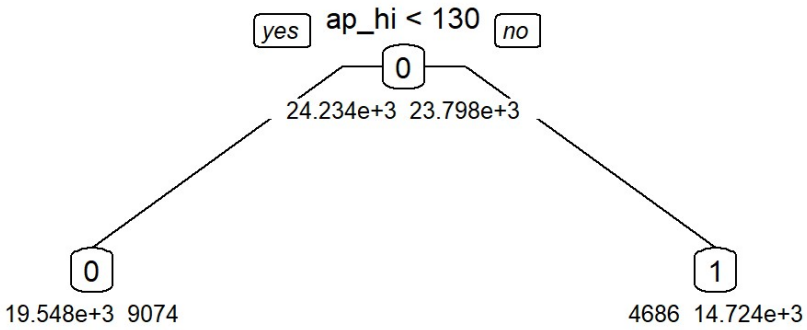
```
## Loading required package: lattice
```

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

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

```
set.seed(1)
cardio.basic.ct <- rpart(cardio ~ ., data = cardio_train, method = "class")

#plot tree
prp(cardio.basic.ct, type = 1, extra = 1, under = TRUE, split.font = 1, varlen = -10, digits =
22)
```



The first rule we get from a default classification tree is IF ap_hi < 130 THEN class = 0 Which translates to if your Systolic blood pressure is less than 130, you don't have cardiovascular disease.

Let's check it's performance on validation set

```
perf.basic.ct <- predict(cardio.basic.ct, newdata = cardio_valid, type = "class")

#confusion matrix
confusionMatrix(perf.basic.ct, cardio_valid$cardio, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 8412 3902
##          1 2023 6249
##
##                Accuracy : 0.7122
##                  95% CI : (0.7059, 0.7184)
##     No Information Rate : 0.5069
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4228
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.6156
##             Specificity : 0.8061
##          Pos Pred Value : 0.7554
##          Neg Pred Value : 0.6831
##              Prevalence : 0.4931
##          Detection Rate : 0.3036
##    Detection Prevalence : 0.4018
##       Balanced Accuracy : 0.7109
##
##        'Positive' Class : 1
##
```

Accuracy: 71.22% Positive Predictive value: 75.54% Sensitivity: 61.56%

Now let's grow a very deep tree with minsplit = 1 and check it's performance

```
set.seed(2)
deep.ct.1 <- rpart(cardio~., data = cardio_train, method = "class", cp = 0, minsplit = 1)
perf.deep.ct.1 <- predict(deep.ct.1, newdata = cardio_valid, type = "class")

#confusion matrix
confusionMatrix(perf.deep.ct.1, cardio_valid$cardio, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 6911 3738
##          1 3524 6413
##
##                Accuracy : 0.6472
##                  95% CI : (0.6407, 0.6538)
##     No Information Rate : 0.5069
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.2941
##  Mcnemar's Test P-Value : 0.01244
##
##             Sensitivity : 0.6318
##             Specificity : 0.6623
##          Pos Pred Value : 0.6454
##          Neg Pred Value : 0.6490
##              Prevalence : 0.4931
##          Detection Rate : 0.3115
##    Detection Prevalence : 0.4827
##       Balanced Accuracy : 0.6470
##
##        'Positive' Class : 1
##
```

Accuracy: 64.72% PPV: 64.54% Sensitivity: 63.18%

Let's prune this deep tree to lowest cp and check performance:

```
set.seed(3)
pruned.low.dt.1 <- prune(deep.ct.1, cp = deep.ct.1$cptable[which.min(deep.ct.1$cptable[,"xerro
r"]),"CP"])
perf.low.dt.1 <- predict(pruned.low.dt.1, newdata = cardio_valid, type = "class")
perf.train.low.dt.1 <- predict(pruned.low.dt.1, cardio_train, type = "class")
confusionMatrix(perf.train.low.dt.1, cardio_train$cardio, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 19115  7086
##          1  5119 16712
##
##                Accuracy : 0.7459
##                  95% CI : (0.742, 0.7498)
##     No Information Rate : 0.5045
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4914
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.7022
##             Specificity : 0.7888
##          Pos Pred Value : 0.7655
##          Neg Pred Value : 0.7296
##              Prevalence : 0.4955
##          Detection Rate : 0.3479
##    Detection Prevalence : 0.4545
##       Balanced Accuracy : 0.7455
##
##        'Positive' Class : 1
##
```

```
confusionMatrix(perf.low.dt.1, cardio_valid$cardio, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 8118 3136
##          1 2317 7015
##
##                Accuracy : 0.7351
##                  95% CI : (0.729, 0.7411)
##     No Information Rate : 0.5069
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4695
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.6911
##             Specificity : 0.7780
##          Pos Pred Value : 0.7517
##          Neg Pred Value : 0.7213
##              Prevalence : 0.4931
##          Detection Rate : 0.3408
##    Detection Prevalence : 0.4533
##       Balanced Accuracy : 0.7345
##
##        'Positive' Class : 1
##
```

Accuracy: 73.51% PPV: 75.17% Sensitivity: 69.11%

Now let's prune more by going up to the best tree.

```
printcp(deep.ct.1)
```

```
##
## Classification tree:
## rpart(formula = cardio ~ ., data = cardio_train, method = "class",
##     cp = 0, minsplit = 1)
##
## Variables actually used in tree construction:
##  [1] active      age         alco        ap_hi       ap_lo
##  [6] bmi         cholesterol gender      gluc        height
## [11] smoke
##
## Root node error: 23798/48032 = 0.49546
##
## n= 48032
##
##            CP nsplit rel error  xerror      xstd
## 1  4.2180e-01      0  1.000000 1.00000 0.0046044
## 2  8.6982e-03      1  0.578200 0.57820 0.0041636
## 3  4.1600e-03      3  0.560803 0.56085 0.0041253
## 4  4.0340e-03      4  0.556643 0.56290 0.0041300
## 5  2.3952e-03      5  0.552609 0.55454 0.0041109
## 6  2.2691e-03      6  0.550214 0.55358 0.0041087
## 7  1.5758e-03      7  0.547945 0.55084 0.0041024
## 8  1.0085e-03      9  0.544794 0.54668 0.0040926
## 9  9.4546e-04     10  0.543785 0.54673 0.0040927
## 10 9.1394e-04     12  0.541894 0.54677 0.0040928
## 11 8.8243e-04     16  0.538239 0.54664 0.0040925
## 12 7.0034e-04     17  0.537356 0.54597 0.0040910
## 13 6.7233e-04     23  0.532524 0.54542 0.0040897
## 14 6.5132e-04     24  0.531851 0.54526 0.0040893
## 15 6.0929e-04     26  0.530549 0.54500 0.0040887
## 16 5.4626e-04     28  0.529330 0.54492 0.0040885
## 17 5.2525e-04     29  0.528784 0.54492 0.0040885
## 18 3.9219e-04     31  0.527733 0.54488 0.0040884
## 19 3.7818e-04     34  0.526557 0.54484 0.0040883
## 20 3.3616e-04     37  0.525422 0.54488 0.0040884
## 21 3.0815e-04     40  0.524330 0.54282 0.0040835
## 22 2.9414e-04     43  0.523405 0.54219 0.0040820
## 23 2.8014e-04     46  0.522523 0.54194 0.0040814
## 24 2.7313e-04     51  0.521052 0.54143 0.0040802
## 25 2.6263e-04     53  0.520506 0.54206 0.0040817
## 26 2.5212e-04     57  0.519455 0.54231 0.0040823
## 27 2.3111e-04     59  0.518951 0.54294 0.0040838
## 28 2.1010e-04     67  0.516934 0.54215 0.0040819
## 29 1.9609e-04     86  0.512858 0.54009 0.0040770
## 30 1.8909e-04     89  0.512270 0.54076 0.0040786
## 31 1.8209e-04     97  0.510379 0.54210 0.0040818
## 32 1.7859e-04    106  0.508698 0.54210 0.0040818
## 33 1.6808e-04    112  0.507606 0.54227 0.0040822
## 34 1.5407e-04    126  0.505169 0.54328 0.0040846
## 35 1.4707e-04    132  0.504202 0.54479 0.0040882
## 36 1.4287e-04    164  0.498781 0.54521 0.0040892
## 37 1.4007e-04    171  0.497605 0.54521 0.0040892
## 38 1.3657e-04    191  0.494537 0.54753 0.0040946
## 39 1.2606e-04    196  0.493823 0.54790 0.0040955
## 40 1.1766e-04    240  0.488066 0.55059 0.0041018
```

```
## 41 1.1556e-04    248  0.487100 0.55110 0.0041030
## 42 1.1345e-04    258  0.485503 0.55110 0.0041030
## 43 1.1205e-04    274  0.483570 0.55122 0.0041032
## 44 1.0505e-04    286  0.481679 0.55299 0.0041073
## 45 1.0272e-04    344  0.475040 0.55379 0.0041092
## 46 9.8047e-05    384  0.469367 0.55467 0.0041112
## 47 9.4546e-05    407  0.467056 0.55538 0.0041129
## 48 9.1044e-05    453  0.460795 0.56034 0.0041242
## 49 8.4041e-05    492  0.456215 0.56038 0.0041243
## 50 7.7037e-05    796  0.429070 0.56492 0.0041345
## 51 7.5637e-05    802  0.428607 0.56744 0.0041401
## 52 7.4703e-05    829  0.426422 0.56744 0.0041401
## 53 7.3536e-05    842  0.425414 0.56870 0.0041429
## 54 7.0034e-05    886  0.421800 0.57001 0.0041458
## 55 6.7233e-05    967  0.415203 0.57799 0.0041632
## 56 6.6032e-05   1007  0.412262 0.57883 0.0041650
## 57 6.3031e-05   1016  0.411631 0.58064 0.0041689
## 58 6.0696e-05   1346  0.388646 0.58686 0.0041821
## 59 5.9529e-05   1364  0.387428 0.58765 0.0041837
## 60 5.8828e-05   1386  0.386041 0.58967 0.0041880
## 61 5.7778e-05   1391  0.385747 0.59181 0.0041924
## 62 5.6027e-05   1416  0.384066 0.59194 0.0041927
## 63 5.4026e-05   1600  0.372006 0.59358 0.0041961
## 64 5.2525e-05   1623  0.370241 0.59673 0.0042025
## 65 5.0424e-05   1740  0.363056 0.59913 0.0042074
## 66 4.9024e-05   1787  0.360450 0.60039 0.0042100
## 67 4.8023e-05   1820  0.358476 0.60039 0.0042100
## 68 4.7273e-05   1837  0.357593 0.60039 0.0042100
## 69 4.6689e-05   1860  0.356206 0.60039 0.0042100
## 70 4.5840e-05   1869  0.355786 0.63127 0.0042696
## 71 4.5022e-05   1972  0.350366 0.63127 0.0042696
## 72 4.2020e-05   1991  0.349399 0.63253 0.0042719
## 73 3.9394e-05   4580  0.234095 0.64039 0.0042862
## 74 3.8519e-05   4655  0.229977 0.64300 0.0042908
## 75 3.8200e-05   4689  0.228549 0.64808 0.0042998
## 76 3.7818e-05   4700  0.228128 0.64867 0.0043008
## 77 3.7351e-05   4836  0.221909 0.64867 0.0043008
## 78 3.6768e-05   4845  0.221573 0.65384 0.0043098
## 79 3.6017e-05   4856  0.221153 0.65396 0.0043100
## 80 3.5017e-05   4992  0.215144 0.65455 0.0043110
## 81 3.3616e-05   5122  0.210186 0.66211 0.0043238
## 82 3.1515e-05   5321  0.202958 0.66426 0.0043274
## 83 3.0560e-05   5846  0.183629 0.67989 0.0043526
## 84 3.0015e-05   5867  0.182830 0.67989 0.0043526
## 85 2.9661e-05   5940  0.180393 0.67989 0.0043526
## 86 2.9414e-05   5971  0.179385 0.68027 0.0043532
## 87 2.8014e-05   6011  0.177872 0.68102 0.0043544
## 88 2.6740e-05   6985  0.146021 0.68468 0.0043601
## 89 2.6263e-05   7007  0.145222 0.68607 0.0043623
## 90 2.5212e-05   7026  0.144718 0.68615 0.0043624
## 91 2.3345e-05   7226  0.139045 0.68640 0.0043628
## 92 2.1010e-05   7237  0.138751 0.71405 0.0044033
## 93 1.9774e-05  10291  0.071224 0.71447 0.0044039
## 94 1.8009e-05  10333  0.070132 0.71548 0.0044053
## 95 1.6808e-05  10416  0.068493 0.71720 0.0044077
```

```
## 96 1.4007e-05  10507  0.066938 0.72153 0.0044136
## 97 1.0505e-05  10897  0.061098 0.72153 0.0044136
## 98 0.0000e+00  10931  0.060677 0.72178 0.0044140
```

```
deep.ct.1$cptable[which.min(deep.ct.1$cptable[,"xerror"]),"CP"]
```

```
## [1] 0.0001960949
```

Tree 29, xerror + xtsd = 0.54009 + 0.0040770 = 0.544167

Tree 27 is the closest with xerror = 0.54294

```
set.seed(4)
pruned.best.dt.1 <- prune(deep.ct.1, cp = 2.3111e-04)
perf.bestpruned.dt.1 <- predict(pruned.best.dt.1, newdata = cardio_valid, type = "class")
confusionMatrix(perf.bestpruned.dt.1, cardio_valid$cardio, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 8134 3129
##          1 2301 7022
##
##                Accuracy : 0.7362
##                  95% CI : (0.7302, 0.7422)
##     No Information Rate : 0.5069
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4718
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.6918
##             Specificity : 0.7795
##          Pos Pred Value : 0.7532
##          Neg Pred Value : 0.7222
##              Prevalence : 0.4931
##          Detection Rate : 0.3411
##    Detection Prevalence : 0.4529
##       Balanced Accuracy : 0.7356
##
##        'Positive' Class : 1
##
```

Accuracy: 73.62% PPV : 75.32% Sensitivity: 69.18% —

Now let's grow a very deep tree with minsplit = 100 and check it's performance

```
set.seed(5)
deep.ct.2 <- rpart(cardio~., data = cardio_train, method = "class", cp = 0, minsplit = 100)
perf.deep.ct.2 <- predict(deep.ct.2, newdata = cardio_valid, type = "class")
perf.train.deep.ct.2 <- predict(deep.ct.2, newdata = cardio_train, type = "class")
confusionMatrix(perf.train.deep.ct.2, cardio_train$cardio, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 19002  6669
##          1  5232 17129
##
##                Accuracy : 0.7522
##                  95% CI : (0.7483, 0.7561)
##     No Information Rate : 0.5045
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5041
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.7198
##             Specificity : 0.7841
##          Pos Pred Value : 0.7660
##          Neg Pred Value : 0.7402
##              Prevalence : 0.4955
##          Detection Rate : 0.3566
##    Detection Prevalence : 0.4655
##       Balanced Accuracy : 0.7519
##
##        'Positive' Class : 1
##
```

```
#confusion matrix
confusionMatrix(perf.deep.ct.2, cardio_valid$cardio, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 7901 3124
##          1 2534 7027
##
##                Accuracy : 0.7252
##                  95% CI : (0.719, 0.7312)
##     No Information Rate : 0.5069
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4498
##  Mcnemar's Test P-Value : 4.863e-15
##
##             Sensitivity : 0.6922
##             Specificity : 0.7572
##          Pos Pred Value : 0.7350
##          Neg Pred Value : 0.7166
##              Prevalence : 0.4931
##          Detection Rate : 0.3413
##    Detection Prevalence : 0.4644
##       Balanced Accuracy : 0.7247
##
##        'Positive' Class : 1
##
```

Accuracy: 72.52% PPV: 73.50% Sensitivity: 69.22%

Pruning this tree by lowest cp

```
set.seed(6)
pruned.low.dt.2 <- prune(deep.ct.2, cp = deep.ct.2$cptable[which.min(deep.ct.2$cptable[,"xerro
r"]),"CP"])
perf.low.dt.2 <- predict(pruned.low.dt.2, newdata = cardio_valid, type = "class")
perf.train.low.dt.2 <- predict(pruned.low.dt.2, newdata = cardio_train, type = "class")
confusionMatrix(perf.train.low.dt.2, cardio_train$cardio, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 18692  6992
##          1  5542 16806
##
##                Accuracy : 0.739
##                  95% CI : (0.7351, 0.743)
##     No Information Rate : 0.5045
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4778
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.7062
##             Specificity : 0.7713
##          Pos Pred Value : 0.7520
##          Neg Pred Value : 0.7278
##              Prevalence : 0.4955
##          Detection Rate : 0.3499
##    Detection Prevalence : 0.4653
##       Balanced Accuracy : 0.7388
##
##        'Positive' Class : 1
##
```

```
confusionMatrix(perf.low.dt.2, cardio_valid$cardio, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 8042 3023
##          1 2393 7128
##
##                Accuracy : 0.7369
##                  95% CI : (0.7308, 0.7429)
##     No Information Rate : 0.5069
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4733
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.7022
##             Specificity : 0.7707
##          Pos Pred Value : 0.7487
##          Neg Pred Value : 0.7268
##              Prevalence : 0.4931
##          Detection Rate : 0.3463
##    Detection Prevalence : 0.4625
##       Balanced Accuracy : 0.7364
##
##        'Positive' Class : 1
##
```

Accuracy : 73.69% PPV: 74.87% Sensitivity: 70.22%

Now pruning the deep tree 2 with best tree

```
printcp(deep.ct.2)
```

```
##
## Classification tree:
## rpart(formula = cardio ~ ., data = cardio_train, method = "class",
##     cp = 0, minsplit = 100)
##
## Variables actually used in tree construction:
##  [1] active      age         ap_hi       ap_lo       bmi
##  [6] cholesterol gender      gluc        height      smoke
##
## Root node error: 23798/48032 = 0.49546
##
## n= 48032
##
##            CP nsplit rel error  xerror      xstd
## 1  4.2180e-01      0   1.00000 1.00000 0.0046044
## 2  8.6982e-03      1   0.57820 0.57820 0.0041636
## 3  4.1600e-03      3   0.56080 0.56097 0.0041256
## 4  4.0340e-03      4   0.55664 0.56106 0.0041258
## 5  2.3952e-03      5   0.55261 0.55303 0.0041074
## 6  2.2691e-03      6   0.55021 0.55211 0.0041053
## 7  1.5758e-03      7   0.54795 0.55055 0.0041017
## 8  1.0085e-03      9   0.54479 0.54710 0.0040936
## 9  9.4546e-04     10   0.54379 0.54526 0.0040893
## 10 9.1394e-04     12   0.54189 0.54534 0.0040895
## 11 8.8243e-04     16   0.53824 0.54526 0.0040893
## 12 7.0034e-04     17   0.53736 0.54492 0.0040885
## 13 6.7233e-04     23   0.53252 0.54551 0.0040899
## 14 6.5132e-04     24   0.53185 0.54513 0.0040890
## 15 6.0929e-04     26   0.53055 0.54484 0.0040883
## 16 5.4626e-04     28   0.52933 0.54551 0.0040899
## 17 5.2525e-04     29   0.52878 0.54408 0.0040865
## 18 3.7818e-04     33   0.52668 0.54286 0.0040836
## 19 3.3616e-04     35   0.52593 0.54429 0.0040870
## 20 2.9414e-04     37   0.52525 0.54442 0.0040873
## 21 2.8014e-04     45   0.52290 0.54496 0.0040886
## 22 2.7313e-04     49   0.52164 0.54521 0.0040892
## 23 2.3111e-04     54   0.52025 0.54635 0.0040918
## 24 2.1010e-04     61   0.51857 0.54677 0.0040928
## 25 1.6808e-04     76   0.51534 0.54635 0.0040918
## 26 1.4707e-04     97   0.51122 0.54807 0.0040959
## 27 1.3657e-04    100   0.51067 0.54853 0.0040970
## 28 1.2606e-04    104   0.51013 0.54929 0.0040987
## 29 1.2006e-04    111   0.50916 0.54984 0.0041000
## 30 1.1556e-04    119   0.50807 0.54984 0.0041000
## 31 1.0505e-04    124   0.50748 0.55110 0.0041030
## 32 1.0085e-04    148   0.50420 0.55303 0.0041074
## 33 9.8047e-05    156   0.50307 0.55299 0.0041073
## 34 8.4041e-05    163   0.50235 0.55631 0.0041150
## 35 7.3536e-05    166   0.50210 0.55660 0.0041157
## 36 6.7233e-05    170   0.50181 0.55698 0.0041165
## 37 6.3031e-05    176   0.50122 0.55782 0.0041184
## 38 5.6027e-05    183   0.50076 0.55790 0.0041186
## 39 4.2020e-05    186   0.50059 0.55841 0.0041198
## 40 2.1010e-05    191   0.50038 0.56261 0.0041293
## 41 1.8676e-05    197   0.50025 0.56303 0.0041302
```

```
## 42 0.0000e+00    206   0.50008 0.56454 0.0041336
```

```
deep.ct.2$cptable[which.min(deep.ct.2$cptable[,"xerror"]),"CP"]
```

```
## [1] 0.000378183
```

Tree 18 is the tree with lowest xerror, xerror + xstd = 0.54286 0.0040836 = 0.5469436 Tree 16 falls within this range with xerror = 0.54551

Using this tree to get best tree

```
set.seed(7)
pruned.best.dt.2 <- prune(deep.ct.2, cp = 5.4626e-04)
perf.bestpruned.dt.2 <- predict(pruned.best.dt.2, newdata = cardio_valid, type = "class")
confusionMatrix(perf.bestpruned.dt.2, cardio_valid$cardio, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 8058 3044
##          1 2377 7107
##
##                Accuracy : 0.7367
##                  95% CI : (0.7306, 0.7427)
##     No Information Rate : 0.5069
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4728
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.7001
##             Specificity : 0.7722
##          Pos Pred Value : 0.7494
##          Neg Pred Value : 0.7258
##              Prevalence : 0.4931
##          Detection Rate : 0.3452
##    Detection Prevalence : 0.4607
##       Balanced Accuracy : 0.7362
##
##        'Positive' Class : 1
##
```

Accuracy : 73.67% PPV : 74.94% Sensitivity: 70.01% —

Now let's grow a very deep tree with minsplit = 500 and check it's performance

```
set.seed(8)
deep.ct.3 <- rpart(cardio~., data = cardio_train, method = "class", cp = 0, minsplit = 500)
perf.deep.ct.3 <- predict(deep.ct.3, newdata = cardio_valid, type = "class")

#confusion matrix
confusionMatrix(perf.deep.ct.3, cardio_valid$cardio, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 8094 3105
##          1 2341 7046
##
##                Accuracy : 0.7355
##                  95% CI : (0.7294, 0.7415)
##     No Information Rate : 0.5069
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4703
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.6941
##             Specificity : 0.7757
##          Pos Pred Value : 0.7506
##          Neg Pred Value : 0.7227
##              Prevalence : 0.4931
##          Detection Rate : 0.3423
##    Detection Prevalence : 0.4560
##       Balanced Accuracy : 0.7349
##
##        'Positive' Class : 1
##
```

Accuracy: 73.55% PPV: 75.06% Sensitivity: 69.41%

Pruning this tree by lowest cp

```
set.seed(9)
pruned.low.dt.3 <- prune(deep.ct.3, cp = deep.ct.3$cptable[which.min(deep.ct.3$cptable[,"xerro
r"]),"CP"])
perf.low.dt.3 <- predict(pruned.low.dt.3, newdata = cardio_valid, type = "class")
confusionMatrix(perf.low.dt.3, cardio_valid$cardio, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 8138 3146
##          1 2297 7005
##
##                Accuracy : 0.7356
##                  95% CI : (0.7295, 0.7416)
##     No Information Rate : 0.5069
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4705
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.6901
##             Specificity : 0.7799
##          Pos Pred Value : 0.7531
##          Neg Pred Value : 0.7212
##              Prevalence : 0.4931
##          Detection Rate : 0.3403
##    Detection Prevalence : 0.4519
##       Balanced Accuracy : 0.7350
##
##        'Positive' Class : 1
##
```

Accuracy: 73.56% PPV : 75.31% Sensitivty: 69.01%

Going up this tree to the best tree

```
printcp(deep.ct.3)
```

```
##
## Classification tree:
## rpart(formula = cardio ~ ., data = cardio_train, method = "class",
##     cp = 0, minsplit = 500)
##
## Variables actually used in tree construction:
## [1] active      age         ap_hi       ap_lo       bmi         cholesterol
## [7] gender      gluc        height
##
## Root node error: 23798/48032 = 0.49546
##
## n= 48032
##
##            CP nsplit rel error  xerror      xstd
## 1  0.42180015      0   1.00000 1.00000 0.0046044
## 2  0.00869821      1   0.57820 0.57820 0.0041636
## 3  0.00416001      3   0.56080 0.56097 0.0041256
## 4  0.00403395      4   0.55664 0.56055 0.0041246
## 5  0.00298344      5   0.55261 0.55736 0.0041174
## 6  0.00226910      6   0.54963 0.55492 0.0041118
## 7  0.00111354      7   0.54736 0.55135 0.0041035
## 8  0.00094546     10   0.54366 0.55068 0.0041020
## 9  0.00091394     12   0.54177 0.54992 0.0041002
## 10 0.00071435     19   0.53475 0.54895 0.0040980
## 11 0.00065132     20   0.53404 0.54799 0.0040957
## 12 0.00033616     22   0.53273 0.54652 0.0040922
## 13 0.00000000     24   0.53206 0.54950 0.0040992
```

```
deep.ct.3$cptable[which.min(deep.ct.3$cptable[,"xerror"]),"CP"]
```

```
## [1] 0.0003361627
```

Tree 12 is with the lowest xerror, xerror + xstd = 0.54652 + 0.0040922 = 0.5506122

Tree 9 comes in this range with xerror = 0.54992

Using this tree to get best tree

```
set.seed(10)
pruned.best.dt.3 <- prune(deep.ct.3, cp = 0.00091394)
perf.bestpruned.dt.3 <- predict(pruned.best.dt.3, newdata = cardio_valid, type = "class")
confusionMatrix(perf.bestpruned.dt.3, cardio_valid$cardio, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 8200 3215
##          1 2235 6936
##
##                Accuracy : 0.7353
##                  95% CI : (0.7292, 0.7413)
##     No Information Rate : 0.5069
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4697
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.6833
##             Specificity : 0.7858
##          Pos Pred Value : 0.7563
##          Neg Pred Value : 0.7184
##              Prevalence : 0.4931
##          Detection Rate : 0.3369
##    Detection Prevalence : 0.4455
##       Balanced Accuracy : 0.7345
##
##        'Positive' Class : 1
##
```

Accuracy : 73.53% PPV: 75.63% Sensitivity: 68.33% —

```
set.seed(11)
deep.ct.4 <- rpart(cardio~., data = cardio_train, method = "class", cp = 0, minsplit = 1000)
perf.deep.ct.4 <- predict(deep.ct.4, newdata = cardio_valid, type = "class")

#confusion matrix
confusionMatrix(perf.deep.ct.4, cardio_valid$cardio, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 8154 3213
##          1 2281 6938
##
##                Accuracy : 0.7331
##                  95% CI : (0.727, 0.7392)
##     No Information Rate : 0.5069
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4655
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.6835
##             Specificity : 0.7814
##          Pos Pred Value : 0.7526
##          Neg Pred Value : 0.7173
##              Prevalence : 0.4931
##          Detection Rate : 0.3370
##    Detection Prevalence : 0.4478
##       Balanced Accuracy : 0.7324
##
##        'Positive' Class : 1
##
```

```
set.seed(12)
pruned.low.dt.4 <- prune(deep.ct.4, cp = deep.ct.4$cptable[which.min(deep.ct.4$cptable[,"xerro
r"]),"CP"])
perf.low.dt.4 <- predict(pruned.low.dt.4, newdata = cardio_valid, type = "class")
confusionMatrix(perf.low.dt.4, cardio_valid$cardio, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 8154 3213
##          1 2281 6938
##
##                Accuracy : 0.7331
##                  95% CI : (0.727, 0.7392)
##     No Information Rate : 0.5069
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4655
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.6835
##             Specificity : 0.7814
##          Pos Pred Value : 0.7526
##          Neg Pred Value : 0.7173
##              Prevalence : 0.4931
##          Detection Rate : 0.3370
##    Detection Prevalence : 0.4478
##       Balanced Accuracy : 0.7324
##
##        'Positive' Class : 1
##
```

```
printcp(deep.ct.4)
```

```
##
## Classification tree:
## rpart(formula = cardio ~ ., data = cardio_train, method = "class",
##     cp = 0, minsplit = 1000)
##
## Variables actually used in tree construction:
## [1] active      age         ap_hi       ap_lo       bmi         cholesterol
## [7] gender      height
##
## Root node error: 23798/48032 = 0.49546
##
## n= 48032
##
##           CP nsplit rel error  xerror      xstd
## 1 0.42180015      0   1.00000 1.00000 0.0046044
## 2 0.00869821      1   0.57820 0.57820 0.0041636
## 3 0.00416001      3   0.56080 0.56110 0.0041259
## 4 0.00403395      4   0.55664 0.56013 0.0041237
## 5 0.00298344      5   0.55261 0.55412 0.0041100
## 6 0.00111354      6   0.54963 0.55202 0.0041051
## 7 0.00094546      8   0.54740 0.54950 0.0040992
## 8 0.00091394     10   0.54551 0.54971 0.0040997
## 9 0.00000000     16   0.53942 0.54668 0.0040926
```

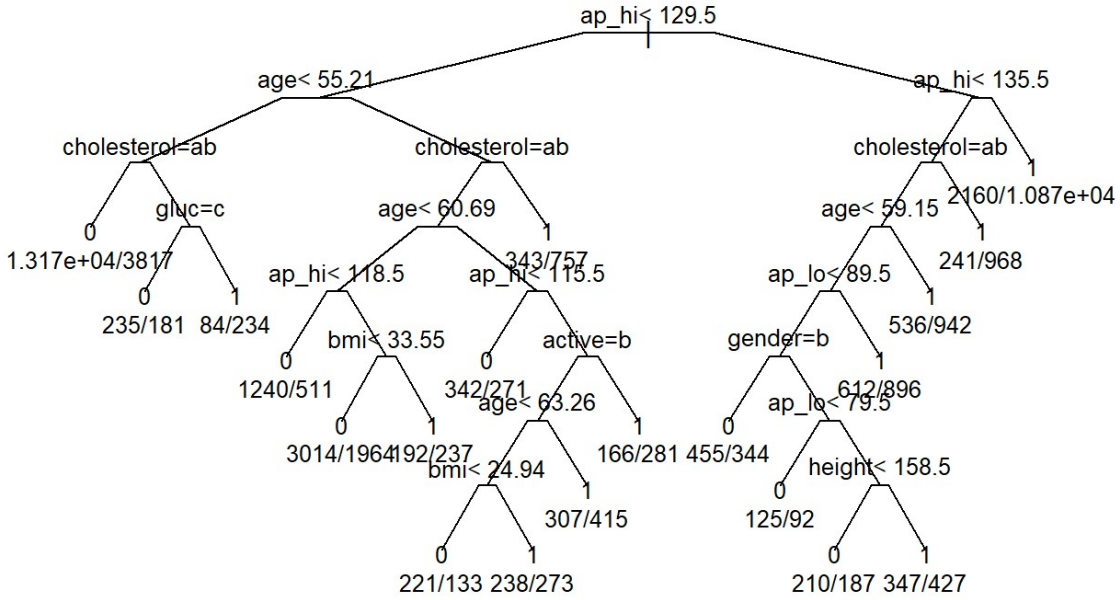
0.55202 + 0.0041051 = 0.5561251 Tree 3 with xerror = 0.56110

```
set.seed(13)
pruned.best.dt.4 <- prune(deep.ct.4, cp = 0.00416001)
perf.bestpruned.dt.4 <- predict(pruned.best.dt.4, newdata = cardio_valid, type = "class")
confusionMatrix(perf.bestpruned.dt.4, cardio_valid$cardio, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 7765 2935
##          1 2670 7216
##
##                Accuracy : 0.7277
##                  95% CI : (0.7216, 0.7338)
##     No Information Rate : 0.5069
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4552
##  Mcnemar's Test P-Value : 0.0004214
##
##             Sensitivity : 0.7109
##             Specificity : 0.7441
##          Pos Pred Value : 0.7299
##          Neg Pred Value : 0.7257
##              Prevalence : 0.4931
##          Detection Rate : 0.3505
##    Detection Prevalence : 0.4802
##       Balanced Accuracy : 0.7275
##
##        'Positive' Class : 1
##
```

Displaying the tree with the best PPV

```
plot(pruned.best.dt.3, uniform = TRUE, compress = TRUE, branch = .2)
text(pruned.best.dt.3, use.n = TRUE, cex = .8, xpd = NA)
```



Variable importance from this tree

```
pruned.best.dt.3$variable.importance
```

```
##       ap_hi       ap_lo cholesterol         age         bmi        gluc
## 5007.863024 2826.871577  978.021175  890.167716  574.156714  224.218390
##      active      height      gender       smoke        alco
##    8.641893    8.571877    6.187303    2.125217    1.617539
```

Removing variables alcohol

```
set.seed(14)
deep.ct.4.noal <- rpart(cardio~., data = cardio_train[c(-8)], method = "class", cp = 0, minspl
it = 1000)
perf.deep.ct.4.noal <- predict(deep.ct.4.noal, newdata = cardio_valid, type = "class")

#confusion matrix
confusionMatrix(perf.deep.ct.4.noal, cardio_valid$cardio, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 8154 3213
##          1 2281 6938
##
##                Accuracy : 0.7331
##                  95% CI : (0.727, 0.7392)
##     No Information Rate : 0.5069
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4655
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.6835
##             Specificity : 0.7814
##          Pos Pred Value : 0.7526
##          Neg Pred Value : 0.7173
##              Prevalence : 0.4931
##          Detection Rate : 0.3370
##    Detection Prevalence : 0.4478
##       Balanced Accuracy : 0.7324
##
##        'Positive' Class : 1
##
```

Accuracy: 73.31% PPV : 75.26%

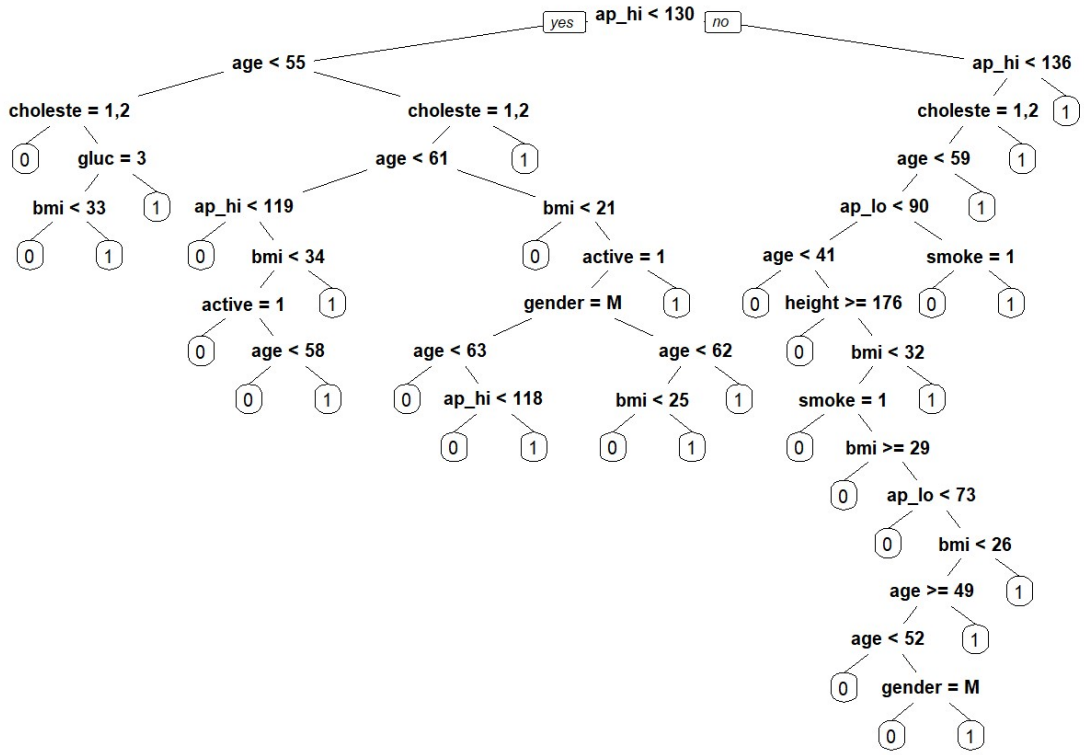
```
set.seed(15)
pruned.low.dt.4.noal <- prune(deep.ct.4.noal, cp = deep.ct.4.noal$cptable[which.min(deep.ct.4.
noal$cptable[,"xerror"]),"CP"])
perf.low.dt.4.noal <- predict(pruned.low.dt.4.noal, newdata = cardio_valid, type = "class")
confusionMatrix(perf.low.dt.4.noal, cardio_valid$cardio, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 7840 2949
##          1 2595 7202
##
##                Accuracy : 0.7307
##                  95% CI : (0.7246, 0.7367)
##     No Information Rate : 0.5069
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.461
##  Mcnemar's Test P-Value : 2.127e-06
##
##             Sensitivity : 0.7095
##             Specificity : 0.7513
##          Pos Pred Value : 0.7351
##          Neg Pred Value : 0.7267
##              Prevalence : 0.4931
##          Detection Rate : 0.3498
##    Detection Prevalence : 0.4759
##       Balanced Accuracy : 0.7304
##
##        'Positive' Class : 1
##
```

Accuracy is decreasing.

Let's display the best tree:

```
prp(pruned.low.dt.2)
```



ROC curve of this tree performance

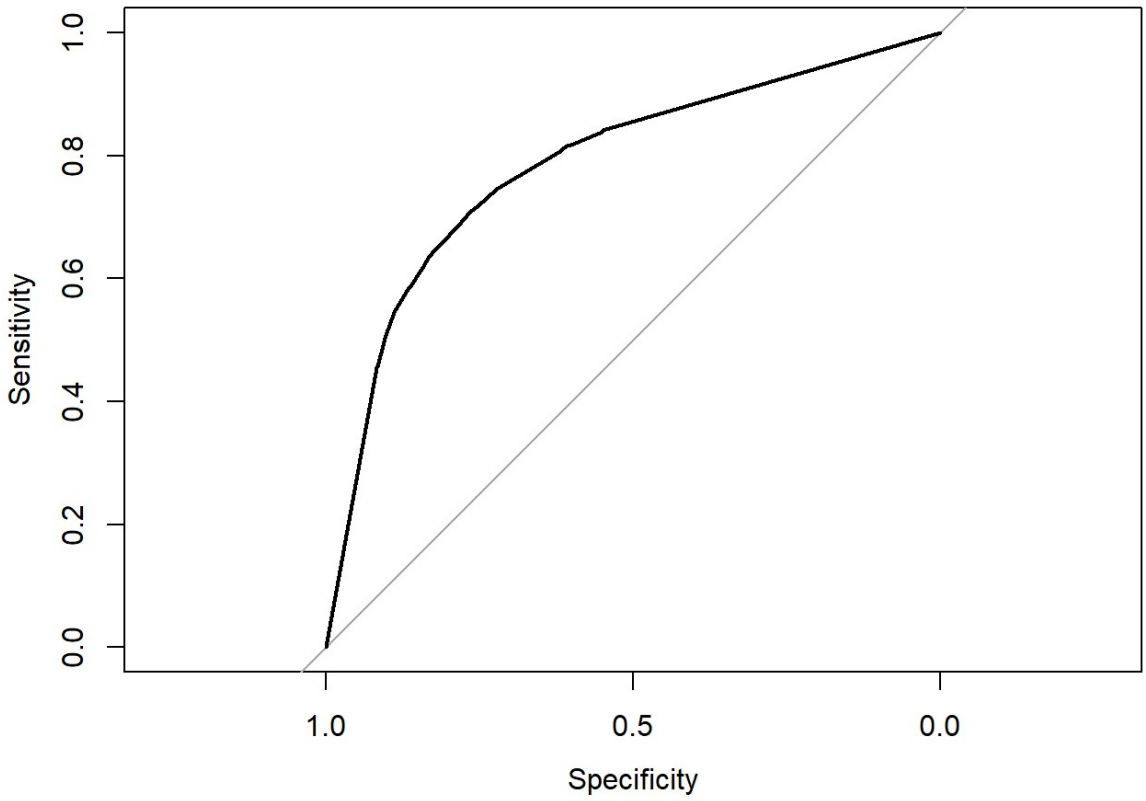
```
library(pROC)
```

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

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

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

```
rocperf <- predict(pruned.low.dt.2, cardio_valid, type = "prob")
rocct <- roc(cardio_valid$cardio, rocperf[,2])
plot.roc(rocct)
```




```
pROC::auc(rocct)
```

```
## Area under the curve: 0.7858
```

```
pruned.low.dt.2$variable.importance
```

```
##       ap_hi       ap_lo cholesterol         age         bmi        gluc
## 5001.333297 2825.060640  978.021175  913.412672  608.901676  224.206274
##      active       smoke      gender      height        alco
##   24.266787   13.407830   11.597163   11.124760    1.983177
```

Removing alcohol variable and running models:

```
deep.ct.2.noalco <- rpart(cardio~., data = cardio_train[,-9], method = "class", cp = 0, minspl
it = 100)
perf.deep.ct.2.noaclo <- predict(deep.ct.2.noalco, newdata = cardio_valid[,-9], type = "clas
s")

#confusion matrix
confusionMatrix(perf.deep.ct.2.noaclo, cardio_valid$cardio, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 7901 3124
##          1 2534 7027
##
##                Accuracy : 0.7252
##                  95% CI : (0.719, 0.7312)
##     No Information Rate : 0.5069
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4498
##  Mcnemar's Test P-Value : 4.863e-15
##
##             Sensitivity : 0.6922
##             Specificity : 0.7572
##          Pos Pred Value : 0.7350
##          Neg Pred Value : 0.7166
##              Prevalence : 0.4931
##          Detection Rate : 0.3413
##    Detection Prevalence : 0.4644
##       Balanced Accuracy : 0.7247
##
##        'Positive' Class : 1
##
```

Accuracy: 72.52 Sensitivity: 69.22 PPV: 73.50

```
pruned.low.dt.2.noalco <- prune(deep.ct.2.noalco, cp = deep.ct.2.noalco$cptable[which.min(dee
p.ct.2.noalco$cptable[,"xerror"]),"CP"])
perf.low.dt.2.noalco <- predict(pruned.low.dt.2.noalco, newdata = cardio_valid[,-9], type = "c
lass")
confusionMatrix(perf.low.dt.2.noalco, cardio_valid$cardio, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 8058 3044
##          1 2377 7107
##
##                Accuracy : 0.7367
##                  95% CI : (0.7306, 0.7427)
##     No Information Rate : 0.5069
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4728
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.7001
##             Specificity : 0.7722
##          Pos Pred Value : 0.7494
##          Neg Pred Value : 0.7258
##              Prevalence : 0.4931
##          Detection Rate : 0.3452
##    Detection Prevalence : 0.4607
##       Balanced Accuracy : 0.7362
##
##        'Positive' Class : 1
##
```

```
printcp(deep.ct.2.noalco)
```

```
##
## Classification tree:
## rpart(formula = cardio ~ ., data = cardio_train[, -9], method = "class",
##     cp = 0, minsplit = 100)
##
## Variables actually used in tree construction:
##  [1] active      age         ap_hi       ap_lo       bmi
##  [6] cholesterol gender      gluc        height      smoke
##
## Root node error: 23798/48032 = 0.49546
##
## n= 48032
##
##            CP nsplit rel error  xerror      xstd
## 1  4.2180e-01      0   1.00000 1.00000 0.0046044
## 2  8.6982e-03      1   0.57820 0.57820 0.0041636
## 3  4.1600e-03      3   0.56080 0.56085 0.0041253
## 4  4.0340e-03      4   0.55664 0.56051 0.0041246
## 5  2.3952e-03      5   0.55261 0.55585 0.0041139
## 6  2.2691e-03      6   0.55021 0.55534 0.0041128
## 7  1.5758e-03      7   0.54795 0.55076 0.0041022
## 8  1.0085e-03      9   0.54479 0.54845 0.0040968
## 9  9.4546e-04     10   0.54379 0.54656 0.0040923
## 10 9.1394e-04     12   0.54189 0.54610 0.0040913
## 11 8.8243e-04     16   0.53824 0.54450 0.0040875
## 12 7.0034e-04     17   0.53736 0.54416 0.0040867
## 13 6.7233e-04     23   0.53252 0.54412 0.0040866
## 14 6.5132e-04     24   0.53185 0.54324 0.0040845
## 15 6.0929e-04     26   0.53055 0.54273 0.0040833
## 16 5.4626e-04     28   0.52933 0.54261 0.0040830
## 17 5.2525e-04     29   0.52878 0.54231 0.0040823
## 18 3.7818e-04     33   0.52668 0.54349 0.0040851
## 19 3.3616e-04     35   0.52593 0.54429 0.0040870
## 20 2.9414e-04     37   0.52525 0.54379 0.0040858
## 21 2.8014e-04     45   0.52290 0.54421 0.0040868
## 22 2.7313e-04     49   0.52164 0.54492 0.0040885
## 23 2.3111e-04     54   0.52025 0.54454 0.0040876
## 24 2.1010e-04     61   0.51857 0.54576 0.0040905
## 25 1.6808e-04     76   0.51534 0.54710 0.0040936
## 26 1.4707e-04     97   0.51122 0.55093 0.0041026
## 27 1.3657e-04    100   0.51067 0.55232 0.0041058
## 28 1.2606e-04    104   0.51013 0.55278 0.0041069
## 29 1.2006e-04    111   0.50916 0.55442 0.0041106
## 30 1.1556e-04    119   0.50807 0.55458 0.0041110
## 31 1.0505e-04    124   0.50748 0.55542 0.0041130
## 32 1.0085e-04    148   0.50420 0.55576 0.0041137
## 33 9.8047e-05    156   0.50307 0.55593 0.0041141
## 34 8.4041e-05    163   0.50235 0.55753 0.0041178
## 35 7.3536e-05    166   0.50210 0.55786 0.0041185
## 36 6.7233e-05    170   0.50181 0.55841 0.0041198
## 37 6.3031e-05    176   0.50122 0.55883 0.0041207
## 38 5.6027e-05    183   0.50076 0.55925 0.0041217
## 39 4.2020e-05    186   0.50059 0.55908 0.0041213
## 40 2.1010e-05    191   0.50038 0.56135 0.0041265
## 41 1.8676e-05    197   0.50025 0.56282 0.0041298
```

```
## 42 0.0000e+00    206   0.50008 0.56496 0.0041346
```

```
deep.ct.2.noalco$cptable[which.min(deep.ct.2.noalco$cptable[,"xerror"]),"CP"]
```

```
## [1] 0.0005252542
```

Tree 17, Best tree : 0.5469035 Tree 9

```
pruned.best.dt.2.noalco<- prune(deep.ct.2.noalco, cp = 0.000945458)
perf.bestpruned.dt.2.noaclo <- predict(pruned.best.dt.2.noalco, newdata = cardio_valid[,-9], t
ype = "class")
confusionMatrix(perf.bestpruned.dt.2.noaclo, cardio_valid$cardio, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 7909 3000
##          1 2526 7151
##
##                Accuracy : 0.7316
##                  95% CI : (0.7255, 0.7376)
##     No Information Rate : 0.5069
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4627
##  Mcnemar's Test P-Value : 1.98e-10
##
##             Sensitivity : 0.7045
##             Specificity : 0.7579
##          Pos Pred Value : 0.7390
##          Neg Pred Value : 0.7250
##              Prevalence : 0.4931
##          Detection Rate : 0.3474
##    Detection Prevalence : 0.4701
##       Balanced Accuracy : 0.7312
##
##        'Positive' Class : 1
##
```

RANDOM FOREST

Let's perform random forest on our data set.

```
set.seed(16)
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
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
cardio.forest <- randomForest(cardio~., data = cardio_train, importance = TRUE)
cardio.forest
```

```
##
## Call:
##  randomForest(formula = cardio ~ ., data = cardio_train, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 27.02%
## Confusion matrix:
##       0     1 class.error
## 0 18722  5512   0.2274490
## 1  7466 16332   0.3137238
```

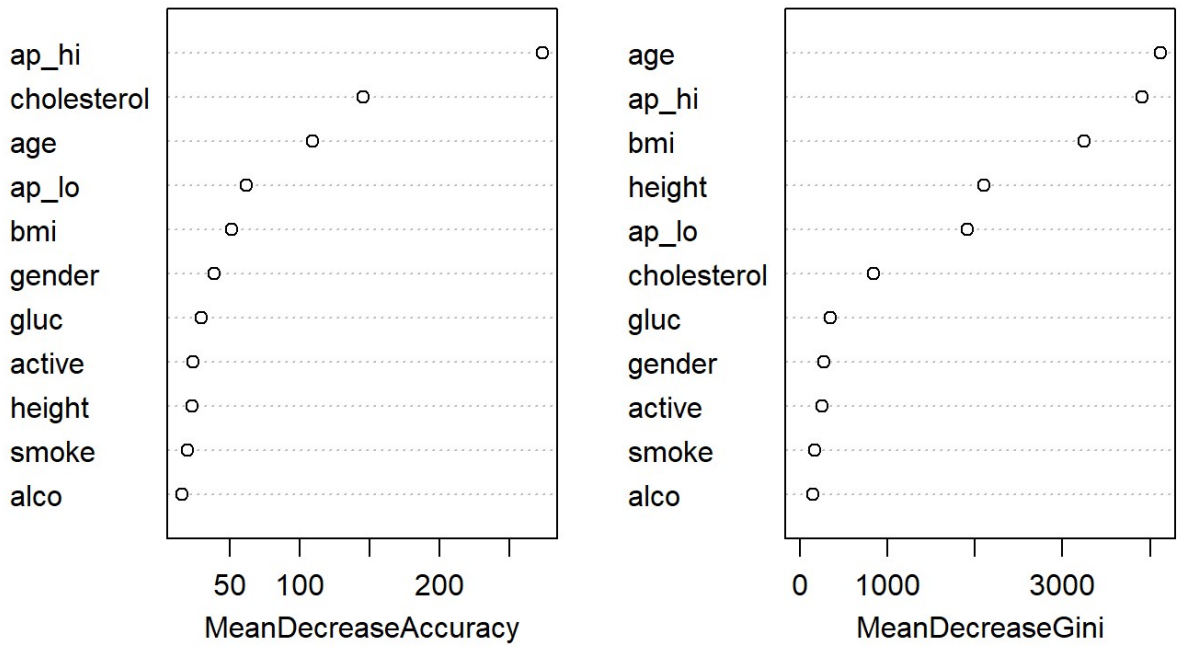
Check prediction performance

```
perf.forest <- predict(cardio.forest, newdata = cardio_valid, type = "class")
confusionMatrix(perf.forest, cardio_valid$cardio, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 8107 3157
##          1 2328 6994
##
##                Accuracy : 0.7336
##                  95% CI : (0.7275, 0.7396)
##     No Information Rate : 0.5069
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4664
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.6890
##             Specificity : 0.7769
##          Pos Pred Value : 0.7503
##          Neg Pred Value : 0.7197
##              Prevalence : 0.4931
##          Detection Rate : 0.3397
##    Detection Prevalence : 0.4528
##       Balanced Accuracy : 0.7330
##
##        'Positive' Class : 1
##
```

variable importance from this random forest

```
varImpPlot(cardio.forest)
```



Random forest with mtry = 2

```
set.seed(17)
cardio.forest.2 <- randomForest(cardio~., data = cardio_train,mtry = 2, importance = TRUE)
cardio.forest.2
```

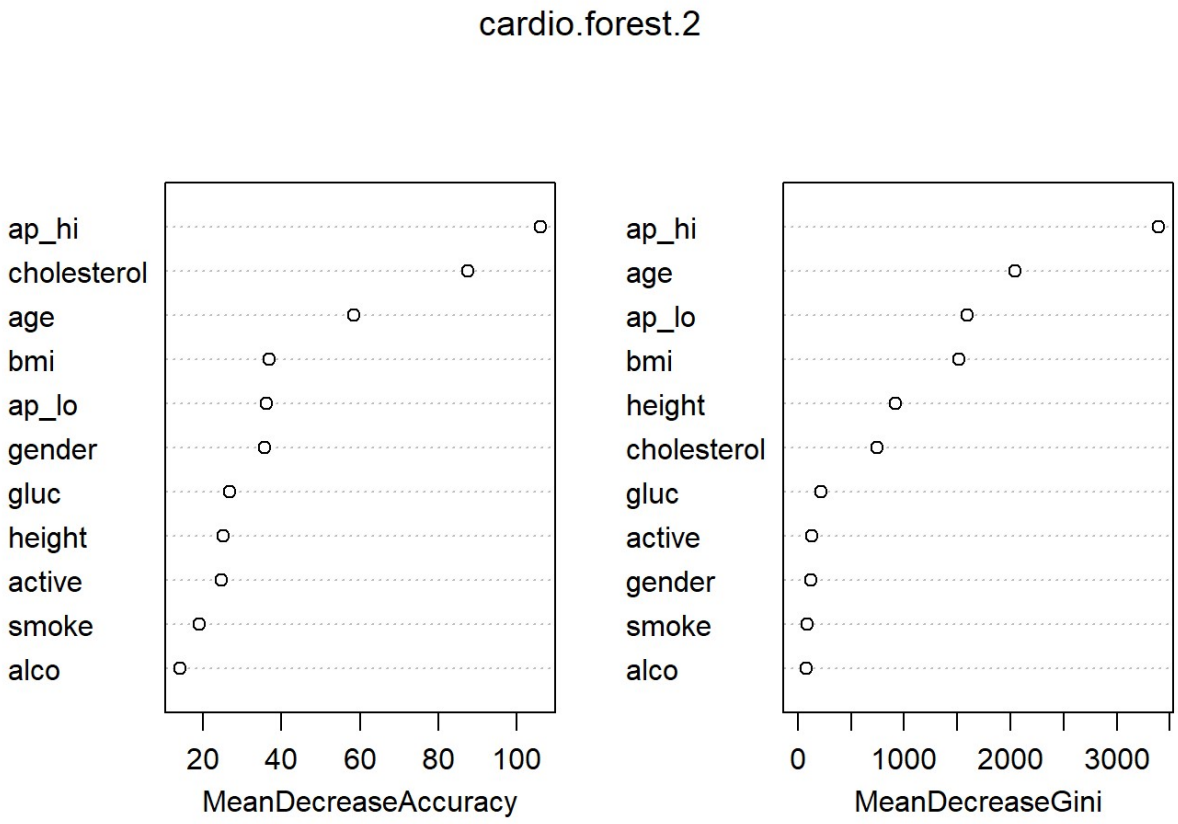
```
##
## Call:
##  randomForest(formula = cardio ~ ., data = cardio_train, mtry = 2,      importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 26.72%
## Confusion matrix:
##       0     1 class.error
## 0 19009  5225   0.2156062
## 1  7607 16191   0.3196487
```

Checking performance

```
perf.forest.2 <- predict(cardio.forest.2, newdata = cardio_valid, type = "class")
confusionMatrix(perf.forest.2, cardio_valid$cardio, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 8237 3235
##          1 2198 6916
##
##                Accuracy : 0.7361
##                  95% CI : (0.73, 0.7421)
##     No Information Rate : 0.5069
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4713
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.6813
##             Specificity : 0.7894
##          Pos Pred Value : 0.7588
##          Neg Pred Value : 0.7180
##              Prevalence : 0.4931
##          Detection Rate : 0.3360
##    Detection Prevalence : 0.4427
##       Balanced Accuracy : 0.7353
##
##        'Positive' Class : 1
##
```

```
varImpPlot(cardio.forest.2)
```



Random forest with mtry = 4

```
set.seed(18)
cardio.forest.3 <- randomForest(cardio~., data = cardio_train,mtry = 4, importance = TRUE)
cardio.forest.3
```

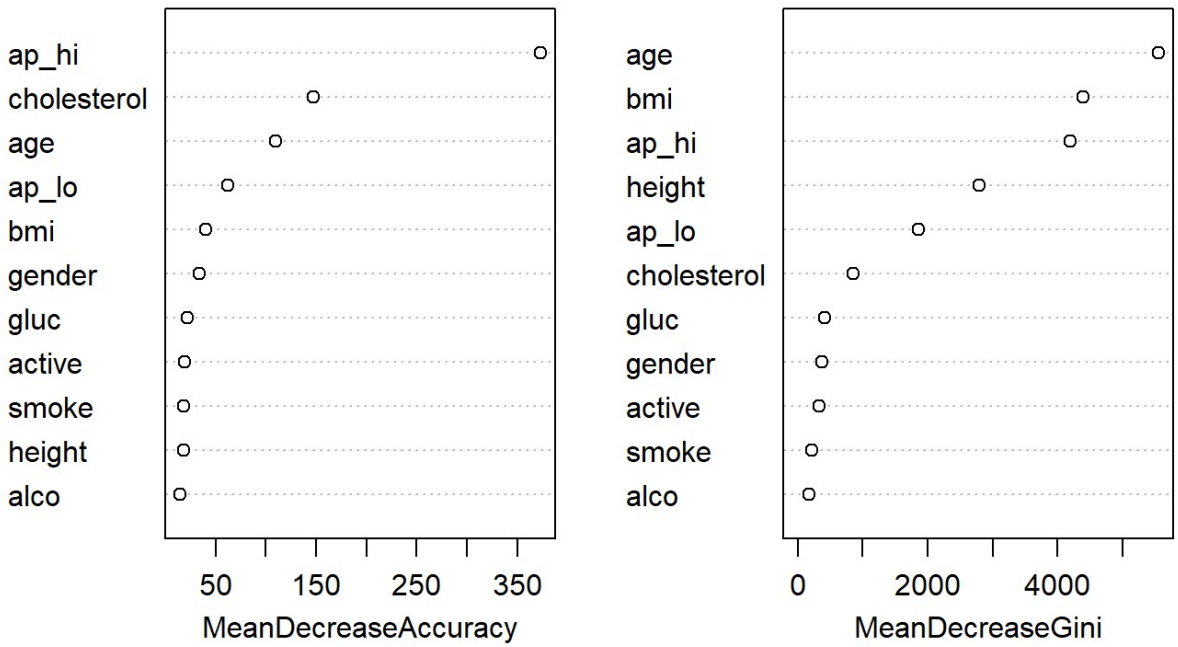
```
##
## Call:
##  randomForest(formula = cardio ~ ., data = cardio_train, mtry = 4,      importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 27.76%
## Confusion matrix:
##       0     1 class.error
## 0 18343  5891   0.2430882
## 1  7444 16354   0.3127994
```

Checking performance

```
perf.forest.3 <- predict(cardio.forest.3, newdata = cardio_valid, type = "class")
confusionMatrix(perf.forest.3, cardio_valid$cardio, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 7999 3131
##          1 2436 7020
##
##                Accuracy : 0.7296
##                  95% CI : (0.7234, 0.7356)
##     No Information Rate : 0.5069
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4585
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.6916
##             Specificity : 0.7666
##          Pos Pred Value : 0.7424
##          Neg Pred Value : 0.7187
##              Prevalence : 0.4931
##          Detection Rate : 0.3410
##    Detection Prevalence : 0.4593
##       Balanced Accuracy : 0.7291
##
##        'Positive' Class : 1
##
```

```
varImpPlot(cardio.forest.3)
```



Random forest without alco variable:

```
set.seed(19)
cardio.forest.2.noalco <-  randomForest(cardio~., data = cardio_train[,-9],mtry = 2, importanc
e = TRUE)
perf.forest.2.noalco <- predict(cardio.forest.2.noalco, cardio_valid[,-9])
confusionMatrix(perf.forest.2.noalco, cardio_valid$cardio, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 8197 3203
##          1 2238 6948
##
##                Accuracy : 0.7357
##                  95% CI : (0.7296, 0.7417)
##     No Information Rate : 0.5069
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4706
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.6845
##             Specificity : 0.7855
##          Pos Pred Value : 0.7564
##          Neg Pred Value : 0.7190
##              Prevalence : 0.4931
##          Detection Rate : 0.3375
##    Detection Prevalence : 0.4462
##       Balanced Accuracy : 0.7350
##
##        'Positive' Class : 1
##
```

Performing Boosted Tree:

```
library(adabag)
```

```
## Loading required package: foreach
```

```
##
## Attaching package: 'foreach'
```

```
## The following objects are masked from 'package:purrr':
##
##     accumulate, when
```

```
## Loading required package: doParallel
```

```
## Loading required package: iterators
```

```
## Loading required package: parallel
```

```
set.seed(20)
boostedtrees <- boosting(cardio~., data = cardio_train)
perf.boosted <- predict(boostedtrees, newdata = cardio_valid, type = "class")
confusionMatrix(as.factor(perf.boosted$class), cardio_valid$cardio, positive = "1")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 8189 3190
##          1 2246 6961
##
##                Accuracy : 0.7359
##                  95% CI : (0.7299, 0.7419)
##     No Information Rate : 0.5069
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4711
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.6857
##             Specificity : 0.7848
##          Pos Pred Value : 0.7561
##          Neg Pred Value : 0.7197
##              Prevalence : 0.4931
##          Detection Rate : 0.3381
##    Detection Prevalence : 0.4472
##       Balanced Accuracy : 0.7353
##
##        'Positive' Class : 1
##
```

Accuracy: 73.59% Positive predictive value: 75.61% Sensitivity: 68.57%