HarvardX: PH125.9x Data Science
Heart Disease Prediction

Battula Damodhar

June 26, 2020

Contents

1 Overview

This project is related to the Heart disease prediction of the HervardX: PH125.9x Data Science: Capstone course. The present report start with a general idea of the project and by representing its objective.

Then the given dataset will be prepared and setup. Data cleaning, Visualization and an exploratory data analysis is carried out in order to develop a machine learning model that could predict dataset. Results will be explained. Finally the report ends with some concluding remarks.

1.1 Introduction

Heart Disease dataset is a dataset available at UCI. In this project we are going to predict the heart disease by using this data set. It is a multivariate dataset with 303 instance and 14 attributes with Catagorical, Real and integer characteristics. This database contains 76 attributes, but all published experiments refer to using a subset of 14 of them. In particular, the Cleveland database is the only one that has been used by ML researchers to this date. The "goal" field refers to the presence of heart disease in the patient. It is integer valued from 0 (no presence) to 4. Experiments with the Cleveland database have concentrated on simply attempting to distinguish presence (values 1,2,3,4) from absence (value 0). The names and social security numbers of the patients were recently removed from the database, replaced with dummy values.

One file has been "processed", that one containing the Cleveland database.

In this project we are going to use this file to predict heart disease.

1.2 Aim of the Project

In this project we are dealing with 3 different machine learning algorithms to predict heart disease (angiographic disease status) are compared. For some algorithms, model parameters are tuned and the best model selected. The best results measured by AUC and accuracy are obtained from a logistic regression model (AUC 0.92, Accuracy 0.87), followed by Gradient Boosting Machines. From a set of 14 variables, the most important to predict heart failure are whether or not there is a reversable defect in Thalassemia followed by whether or not there is an occurrence of asymptomatic chest pain.

1.3 Dataset

Nicely prepared heart disease data are available at UCI. The description of the database can be found here. The document mentions that previous work resulted in an accuracy of 74-77% for the preciction of heart disease using the cleveland data.

Variable name	Short desciption	Variable name	Short description
age	Age of patient	thalach	maximum heart rate achieved
sex	Sex, 1 for male	exang	exercise induced angina (1 yes)
cp	chest pain	oldpeak	ST depression induc. ex.
trestbps	resting blood pressure	slope	slope of peak exercise ST
chol	serum cholesterol	ca	number of major vessel
fbs	fasting blood sugar larger 120mg/dl (1 true)	thal	no explanation provided, but probably thalassemia (3 normal; 6 fixed defect; 7 reversable defect)
restecg	resting electroc. result (1 anomality)	num	diagnosis of heart disease (angiographic disease status)

The variable we want to predict is **num** with Value 0: < 50% diameter narrowing and Value 1: > 50% diameter narrowing. We assume that every value with 0 means heart is okay, and 1,2,3,4 means heart disease.

From the possible values the variables can take, it is evident that the following need to be dummified because the distances in the values is random: cp,thal, restecg, slope

Function needed to convert classes of predictor values.

```
convert_magic <- function(obj,types){
    for (i in 1:length(obj)){
        FUN <- switch(types[i],character = as.character,
                                   numeric = as.numeric,
                                   factor = as.factor)
        obj[,i] <- FUN(obj[,i])
    }
    obj
}
```

This report has been created thanks to an automatised report (R markdown). The whole code and work can be found at : https://github.com/DamodharB/MyProject/tree/master/Capstone_Project/MovieLense_Project

Our work is broken down into several parts :
1. Data pre-processing
2. Attribute informations
3. Data Visualization
4. Methods and Analysis
4.1 Data cleaning
4.2 Data analysis
4.3 Modelling Approach

2 Data pre-processing

Nicely prepared heart disease data are available at UCI The description of the database can be found here. We first import datasets from this site in order to load them as dataframe. Then, we add a header to those dataframes.

```r
process_data <- function(data_source){
  data <- read.table(data_source, sep = ',', na = '?', header = FALSE)
  colnames(data) <- c("age","sex","cp","trestbps","chol","fbs","restecg","thalach","exang","oldpeak","sl
  return(data)
}

cleveland_data <- process_data('https://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/
```

Get a quick idea of the data:

```r
dim(cleveland_data)
```

```
## [1] 303  14
```

```r
head(cleveland_data)
```

```
##   age sex cp trestbps chol fbs restecg thalach exang oldpeak slope ca thal num
## 1  63   1  1      145  233   1       2     150     0     2.3     3  0    6   0
## 2  67   1  4      160  286   0       2     108     1     1.5     2  3    3   2
## 3  67   1  4      120  229   0       2     129     1     2.6     2  2    7   1
## 4  37   1  3      130  250   0       0     187     0     3.5     3  0    3   0
## 5  41   0  2      130  204   0       2     172     0     1.4     1  0    3   0
## 6  56   1  2      120  236   0       0     178     0     0.8     1  0    3   0
```

3 Attribute Informations

```r
str(cleveland_data)
```

```
## 'data.frame':    303 obs. of  14 variables:
##  $ age     : num  63 67 67 37 41 56 62 57 63 53 ...
##  $ sex     : num  1 1 1 1 0 1 0 0 1 1 ...
##  $ cp      : num  1 4 4 3 2 2 4 4 4 4 ...
##  $ trestbps: num  145 160 120 130 130 120 140 120 130 140 ...
##  $ chol    : num  233 286 229 250 204 236 268 354 254 203 ...
##  $ fbs     : num  1 0 0 0 0 0 0 0 0 1 ...
##  $ restecg : num  2 2 2 0 2 0 2 0 2 2 ...
##  $ thalach : num  150 108 129 187 172 178 160 163 147 155 ...
##  $ exang   : num  0 1 1 0 0 0 0 1 0 1 ...
##  $ oldpeak : num  2.3 1.5 2.6 3.5 1.4 0.8 3.6 0.6 1.4 3.1 ...
##  $ slope   : num  3 2 2 3 1 1 3 1 2 3 ...
##  $ ca      : num  0 3 2 0 0 0 2 0 1 0 ...
##  $ thal    : num  6 3 7 3 3 3 3 3 7 7 ...
##  $ num     : int  0 2 1 0 0 0 3 0 2 1 ...
```

We have total 303 objects and 14 attributes for the cleveland dataset.

Firstly, all attributes are numerical except num which is an int attribute. It's the goal attribute (the one we are trying to guess), the diagnosis of heart disease.

We notice that age, trestbps, chol, thalach and oldpeak are continuous variables while other attributes have categorical values (they were in original dataset represented as string categorical values but they were mapped as numerical values to compute more easily).

```
summary(cleveland_data)
```

```
##       age              sex               cp           trestbps    
##  Min.   :29.00   Min.   :0.0000   Min.   :1.000   Min.   : 94.0  
##  1st Qu.:48.00   1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:120.0  
##  Median :56.00   Median :1.0000   Median :3.000   Median :130.0  
##  Mean   :54.44   Mean   :0.6799   Mean   :3.158   Mean   :131.7  
##  3rd Qu.:61.00   3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:140.0  
##  Max.   :77.00   Max.   :1.0000   Max.   :4.000   Max.   :200.0  
##                                                                  
##       chol            fbs            restecg          thalach     
##  Min.   :126.0   Min.   :0.0000   Min.   :0.0000   Min.   : 71.0  
##  1st Qu.:211.0   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:133.5  
##  Median :241.0   Median :0.0000   Median :1.0000   Median :153.0  
##  Mean   :246.7   Mean   :0.1485   Mean   :0.9901   Mean   :149.6  
##  3rd Qu.:275.0   3rd Qu.:0.0000   3rd Qu.:2.0000   3rd Qu.:166.0  
##  Max.   :564.0   Max.   :1.0000   Max.   :2.0000   Max.   :202.0  
##                                                                   
##      exang            oldpeak         slope             ca        
##  Min.   :0.0000   Min.   :0.00   Min.   :1.000   Min.   :0.0000  
##  1st Qu.:0.0000   1st Qu.:0.00   1st Qu.:1.000   1st Qu.:0.0000  
##  Median :0.0000   Median :0.80   Median :2.000   Median :0.0000  
##  Mean   :0.3267   Mean   :1.04   Mean   :1.601   Mean   :0.6722  
##  3rd Qu.:1.0000   3rd Qu.:1.60   3rd Qu.:2.000   3rd Qu.:1.0000  
##  Max.   :1.0000   Max.   :6.20   Max.   :3.000   Max.   :3.0000  
##                                                  NA's   :4       
##       thal            num        
##  Min.   :3.000   Min.   :0.0000  
##  1st Qu.:3.000   1st Qu.:0.0000  
##  Median :3.000   Median :0.0000  
##  Mean   :4.734   Mean   :0.9373  
##  3rd Qu.:7.000   3rd Qu.:2.0000  
##  Max.   :7.000   Max.   :4.0000  
##  NA's   :2
```

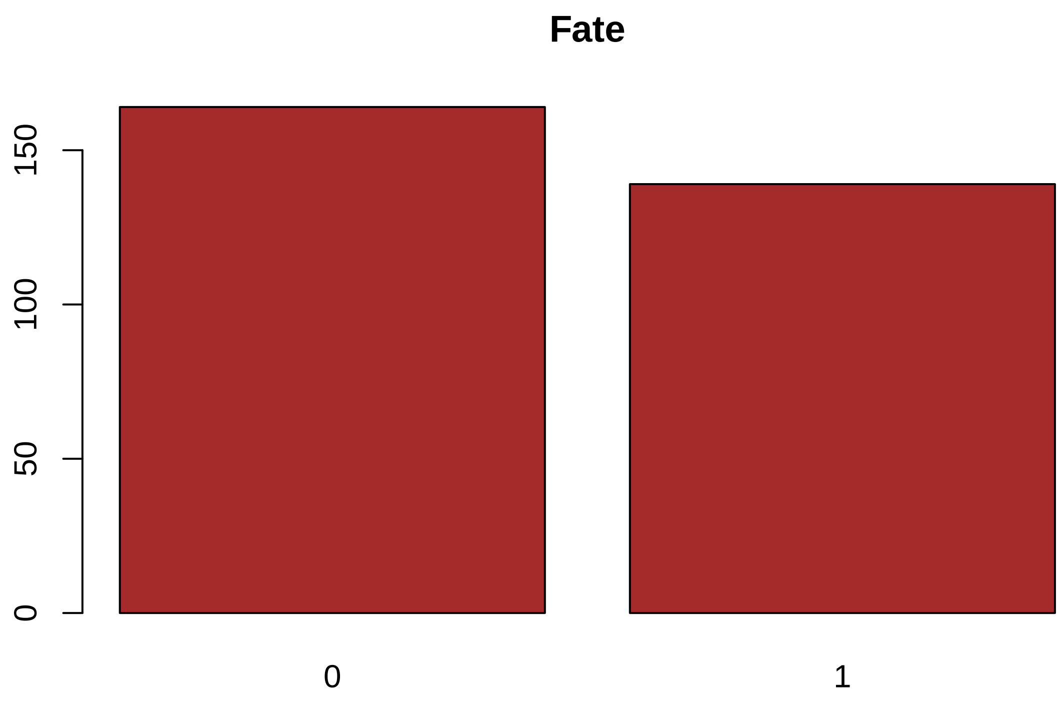
We noticed that ca and thal attributes in cleveland_data dataset having NA values.

4 Data visualization

Explore the data quickly, how many had heart attack, women or men, age?

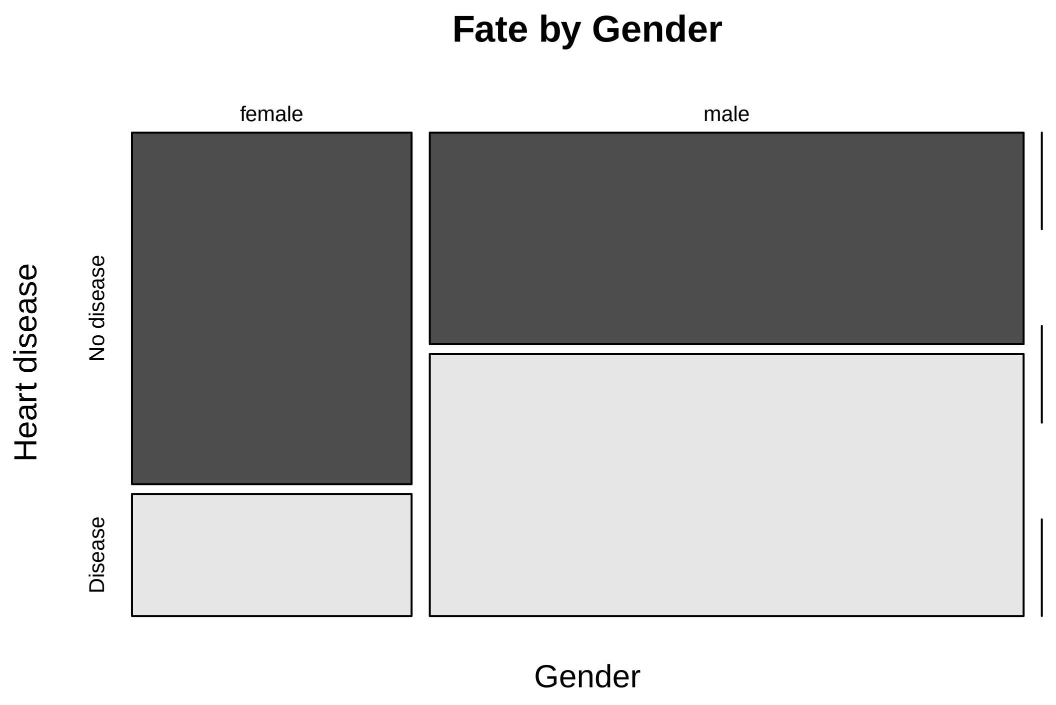
Values of num > 0 are cases of heart disease. Dummify some variables.

```
cleveland_data$num[cleveland_data$num > 0] <- 1
barplot(table(cleveland_data$num), main="Fate", col="brown")
```

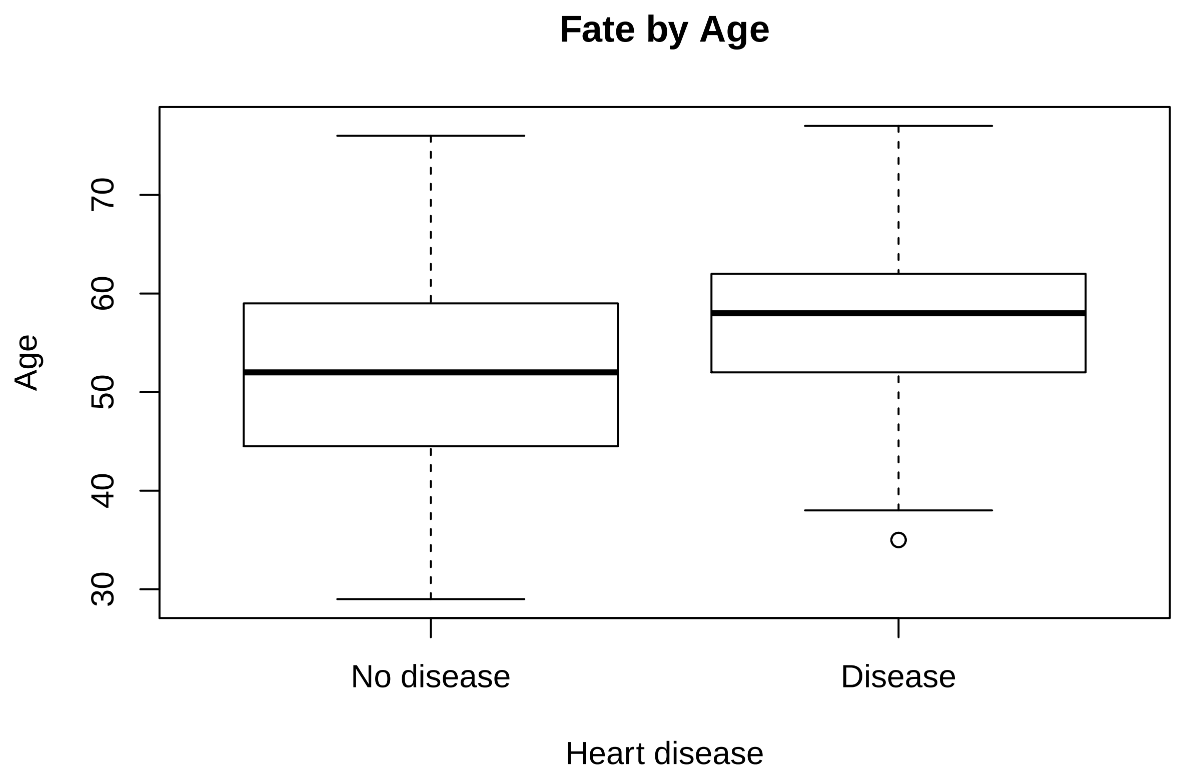


change a few predictor variables from integer to factors (make dummies).

```
chclass <-c("numeric","factor","factor","numeric","numeric","factor","factor","numeric","factor","numer

cleveland_data1 <- convert_magic(cleveland_data,chclass)

heart = cleveland_data1 #add labels only for plot
levels(heart$num) = c("No disease","Disease")
levels(heart$sex) = c("female","male","")
mosaicplot(heart$sex ~ heart$num,
           main="Fate by Gender", shade=FALSE,color=TRUE,
           xlab="Gender", ylab="Heart disease")
```

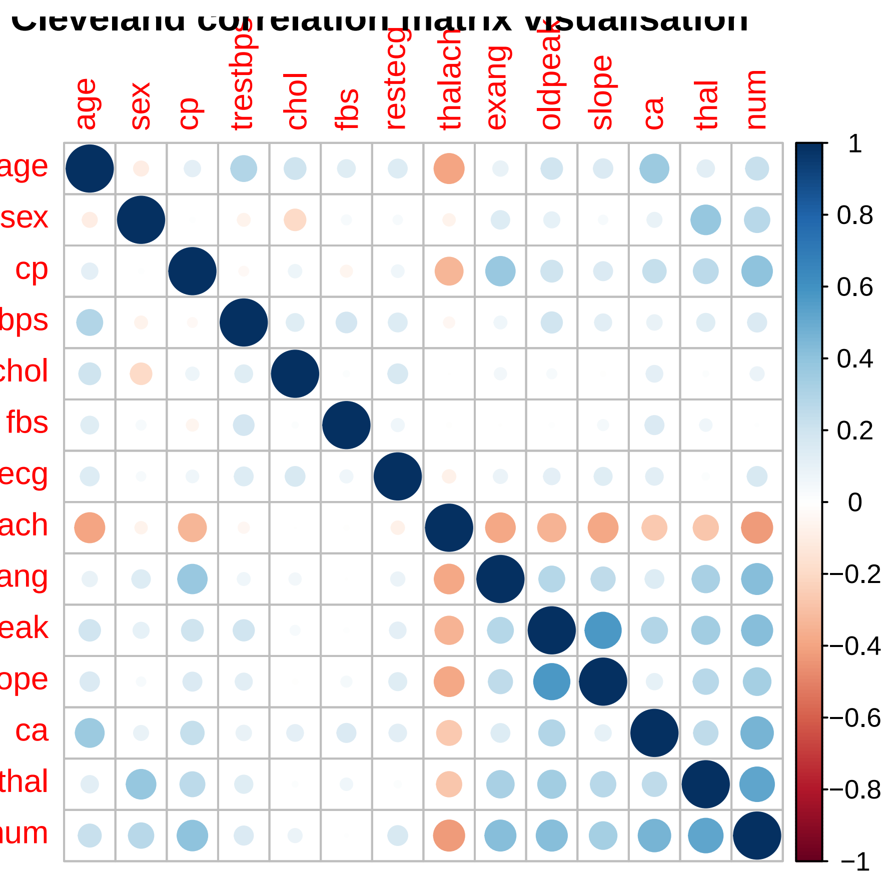


```r
boxplot(heart$age ~ heart$num,
        main="Fate by Age",
        ylab="Age",xlab="Heart disease")
```



We install the corrplot package to visualize our correlation matrix easily.

```r
library(corrplot)

corr_matrix <- function(dataframe, t){
  df_clean <- na.omit(dataframe)
  cat("\n")
  corrplot(cor(df_clean), method = "circle", title = t)
}

corr_matrix(cleveland_data, "Cleveland correlation matrix visualisation")
```



For cleveland dataframe we want to replace NA values from 2 discrete attributes, CA and THAL. We can see on the correlation plot that: - CA attibute is mostly correlated to NUM and AGE attribute, - THAL attribute is most strongly correlated to SEX and NUM attributes.

5 Methods and Analysis

5.1 Data cleaning

Missing values in data is a common phenomenon in real world problems. Knowing how to handle missing values effectively is a required step to reduce bias and to produce powerful models. Lets explore various options of how to deal with missing values and how to implement them. We can handle the missing data using different techniques depending if there is a lot of unvalued data or not. If not, we can delete NA marked rows or we can replace missing data by the mean of the associated attribute (column) for continuous values. Otherwise, we could use more advanced technical stuff like logistic regression to deduce discretes values (for instance 1, 2, 3 or 4 as possible outputs for one attribute) or linear regressions for continuous values. We will also compute a correlation matrix to know which attributes influence the one where the value is missing so that we could deduce it. We could finally have simply delete those rows but that would have mean losing information (even if it's a small part).

Check for missing values - only 6 so just remove them.

```
s = sum(is.na(cleveland_data))
cleveland_data_clean <- na.omit(cleveland_data)

cat("CLEVELAND DATAFRAME\n")
```

```
## CLEVELAND DATAFRAME
```

```
summary(cleveland_data_clean)
```

```
##       age             sex               cp           trestbps    
##  Min.   :29.00   Min.   :0.0000   Min.   :1.000   Min.   : 94.0  
##  1st Qu.:48.00   1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:120.0  
##  Median :56.00   Median :1.0000   Median :3.000   Median :130.0  
##  Mean   :54.54   Mean   :0.6768   Mean   :3.158   Mean   :131.7  
##  3rd Qu.:61.00   3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:140.0  
##  Max.   :77.00   Max.   :1.0000   Max.   :4.000   Max.   :200.0  
##       chol            fbs            restecg           thalach     
##  Min.   :126.0   Min.   :0.0000   Min.   :0.0000   Min.   : 71.0  
##  1st Qu.:211.0   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:133.0  
##  Median :243.0   Median :0.0000   Median :1.0000   Median :153.0  
##  Mean   :247.4   Mean   :0.1448   Mean   :0.9966   Mean   :149.6  
##  3rd Qu.:276.0   3rd Qu.:0.0000   3rd Qu.:2.0000   3rd Qu.:166.0  
##  Max.   :564.0   Max.   :1.0000   Max.   :2.0000   Max.   :202.0  
##      exang           oldpeak          slope             ca        
##  Min.   :0.0000   Min.   :0.000   Min.   :1.000   Min.   :0.0000  
##  1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:1.000   1st Qu.:0.0000  
##  Median :0.0000   Median :0.800   Median :2.000   Median :0.0000  
##  Mean   :0.3266   Mean   :1.056   Mean   :1.603   Mean   :0.6768  
##  3rd Qu.:1.0000   3rd Qu.:1.600   3rd Qu.:2.000   3rd Qu.:1.0000  
##  Max.   :1.0000   Max.   :6.200   Max.   :3.000   Max.   :3.0000  
##       thal            num        
##  Min.   :3.000   Min.   :0.0000  
##  1st Qu.:3.000   1st Qu.:0.0000  
##  Median :3.000   Median :0.0000  
##  Mean   :4.731   Mean   :0.4613  
##  3rd Qu.:7.000   3rd Qu.:1.0000  
##  Max.   :7.000   Max.   :1.0000
```

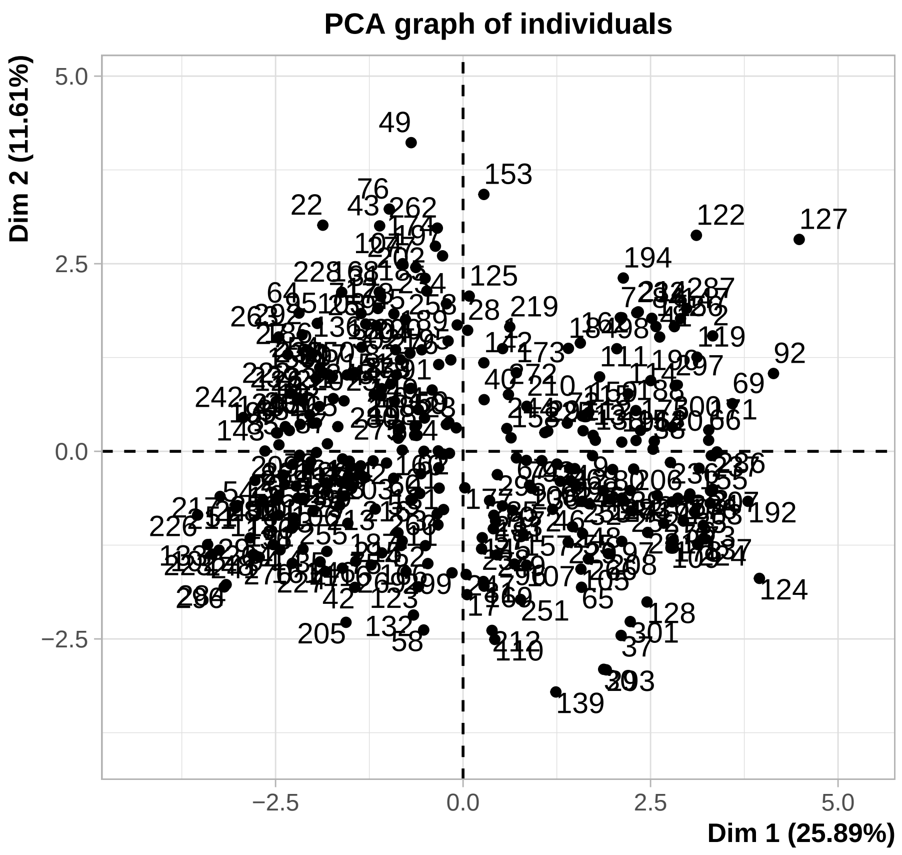
Now the cleveland dataset is fully cleaned.

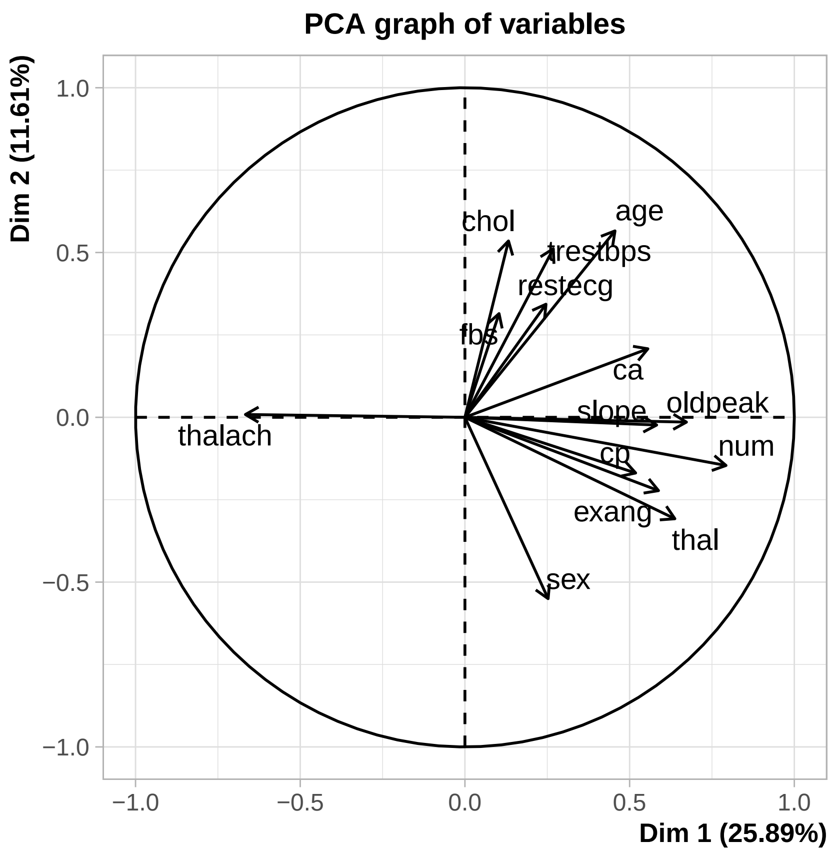
5.2 Data Analysis

Once dataframe has been cleaned, we begin first by interpreting how attributes are correlated to num attribute (attribute in which we would like to predict later).

As we have a lot of attributes to analyze, comparing each other by hand will be too long and not efficient. It is better to see how each attribute are linked by using Principal Component Analysis method.

```
library(FactoMineR)

cleveland_dataframe = as.data.frame(cleveland_data_clean)
result = PCA(cleveland_dataframe[,1:14])
```


PCA graph of individuals
Dim 1 (25.89%)
Dim 2 (11.61%)
−2.5
0.0
2.5
5.0
−2.5
0.0
2.5
5.0
49
153
122
127
194
124
139
92
69
205
22
76
58
110



As we can see in this correlation circle, it appears that the main attributes that impact on heart disease (num attribute) could be the attributes oldpeak, thal, cp, exang and slope. One attribute is negatively correlated to num which is thalach. All the attributes that are correlated (negatively or positively) to Dim 2 are mainly about heart statement. The other attributes like age, sex, trestbps, chol, fbs and restecg seems not to be highly correlated to heart disease. This time, all the attributes that are correlated to Dim 1 are mainly about blood statement. Even if we lost some information, PCA allows us to better understand different attributes and how they can be related.

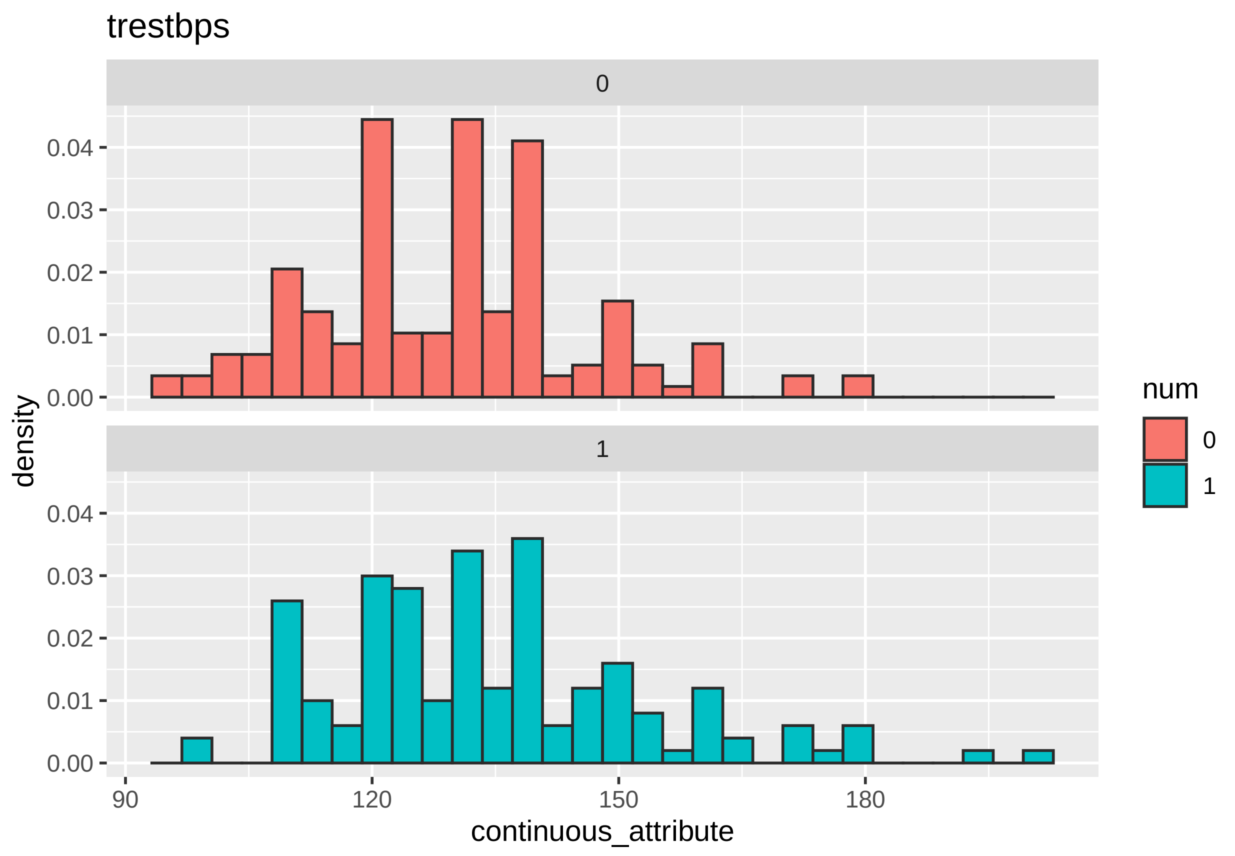
Now, we will print an histogram so that we can analyze which of these attributes can be correlated to heart disease. But before that, we have to convert categorical values into vector so that we can print their histogram depending on num.

```
cleveland_dataframe$num <- as.factor(cleveland_dataframe$num)
cleveland_dataframe$sex <- as.factor(cleveland_dataframe$sex)
cleveland_dataframe$cp <- as.factor(cleveland_dataframe$cp)
cleveland_dataframe$restecg <- as.factor(cleveland_dataframe$restecg)
cleveland_dataframe$fbs <- as.factor(cleveland_dataframe$fbs)
cleveland_dataframe$thal <- as.factor(cleveland_dataframe$thal)
cleveland_dataframe$ca <- as.factor(cleveland_dataframe$ca)
cleveland_dataframe$slope <- as.factor(cleveland_dataframe$slope)
```

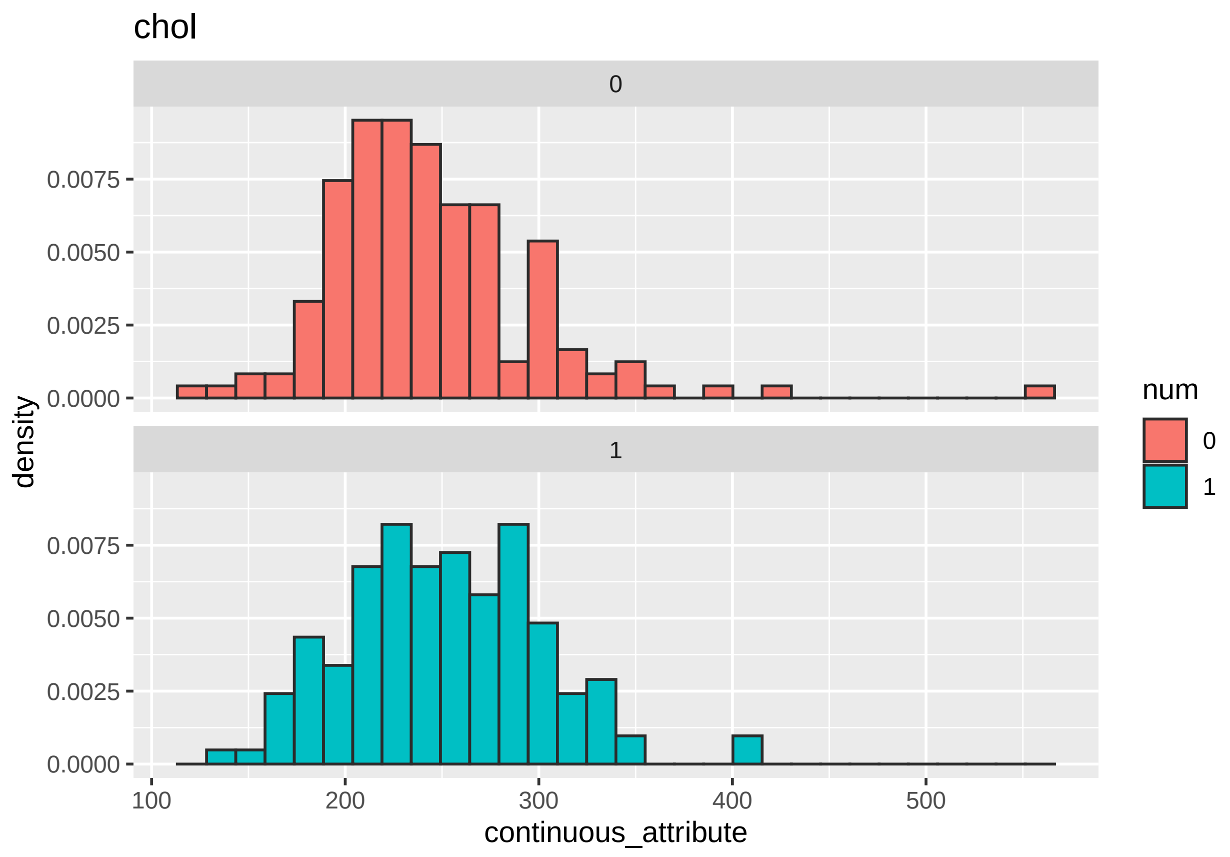
```
create_level <- function(city_categorical_attributes,categorical_attribute,number_of_category){
  i=0
  for (i in number_of_category){
    levels(city_categorical_attributes$categorical_attribute)[levels(city_categorical_attributes$catego
  }
}

create_level(cleveland_dataframe,sex,1)
create_level(cleveland_dataframe,fbs,1)
create_level(cleveland_dataframe,thal,3)
create_level(cleveland_dataframe,num,4)
create_level(cleveland_dataframe,exang,1)
create_level(cleveland_dataframe,cp,3)
create_level(cleveland_dataframe,restecg,2)
create_level(cleveland_dataframe,slope,2)
```

Once each categorical value has been converted into vector, we can plot each attribute :

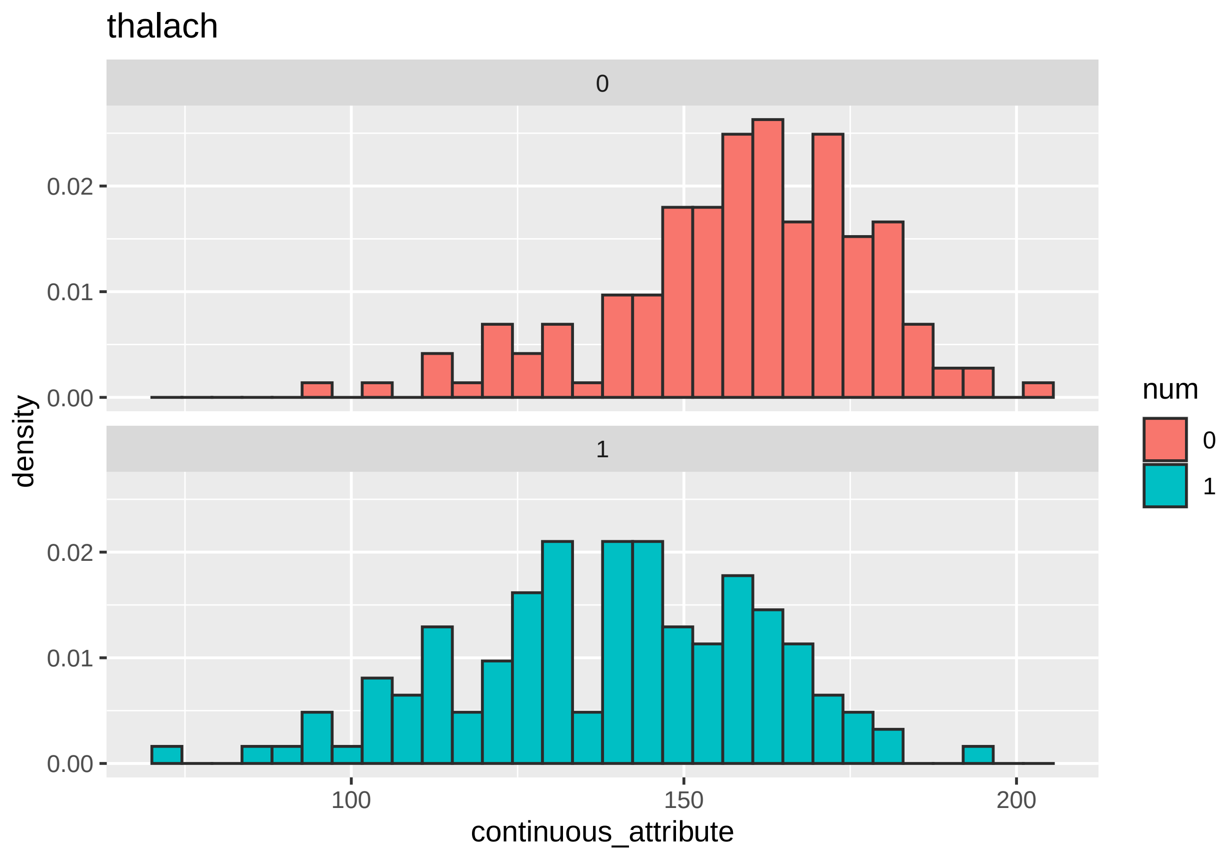
```
library(ggplot2)
library(ggthemes)

plot_histogram <- function(city_dataframe, continuous_attribute,title_plot){
  ggplot(data = city_dataframe,mapping = aes(x = continuous_attribute, fill=num)) + geom_histogram(aes(
    facet_wrap(~num, ncol=1,scale="fixed") + ggtitle(title_plot)
}
plot_bar <- function(city_dataframe, categorical_attribute,title_plot){
  ggplot(data = city_dataframe, mapping = aes(x = categorical_attribute)) + geom_bar() + facet_wrap(~nu
}

plot_histogram(cleveland_dataframe,cleveland_dataframe$trestbps,"trestbps")
```



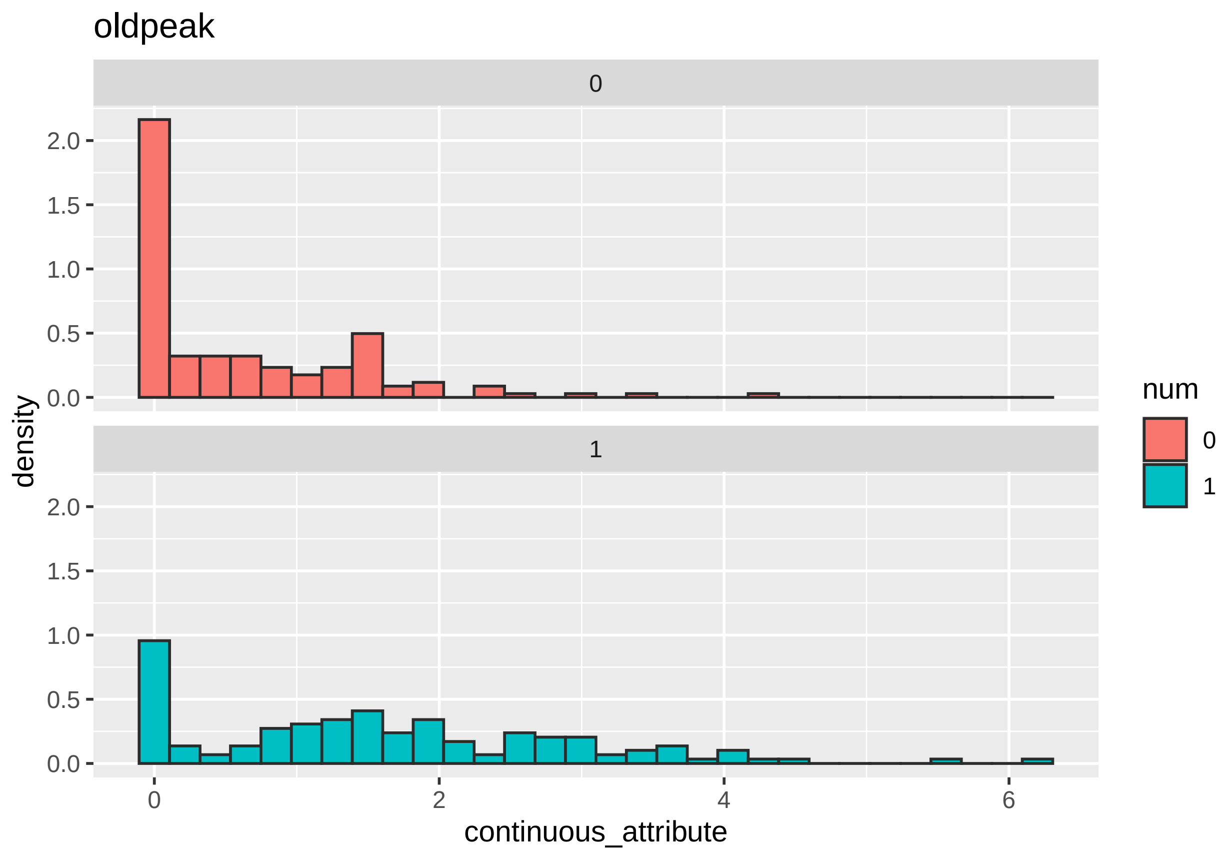
```r
plot_histogram(cleveland_dataframe,cleveland_dataframe$chol,"chol")
```



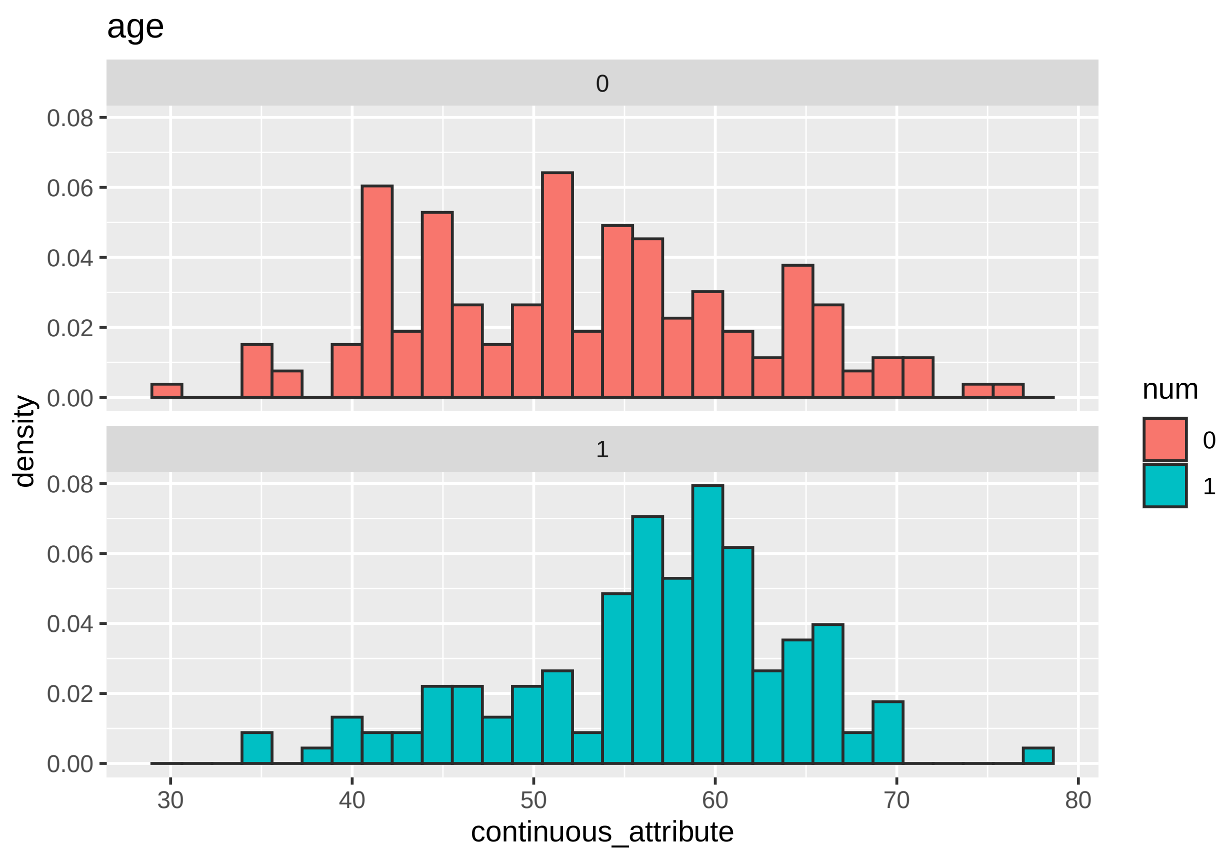
```r
plot_histogram(cleveland_dataframe,cleveland_dataframe$thalach,"thalach")
```



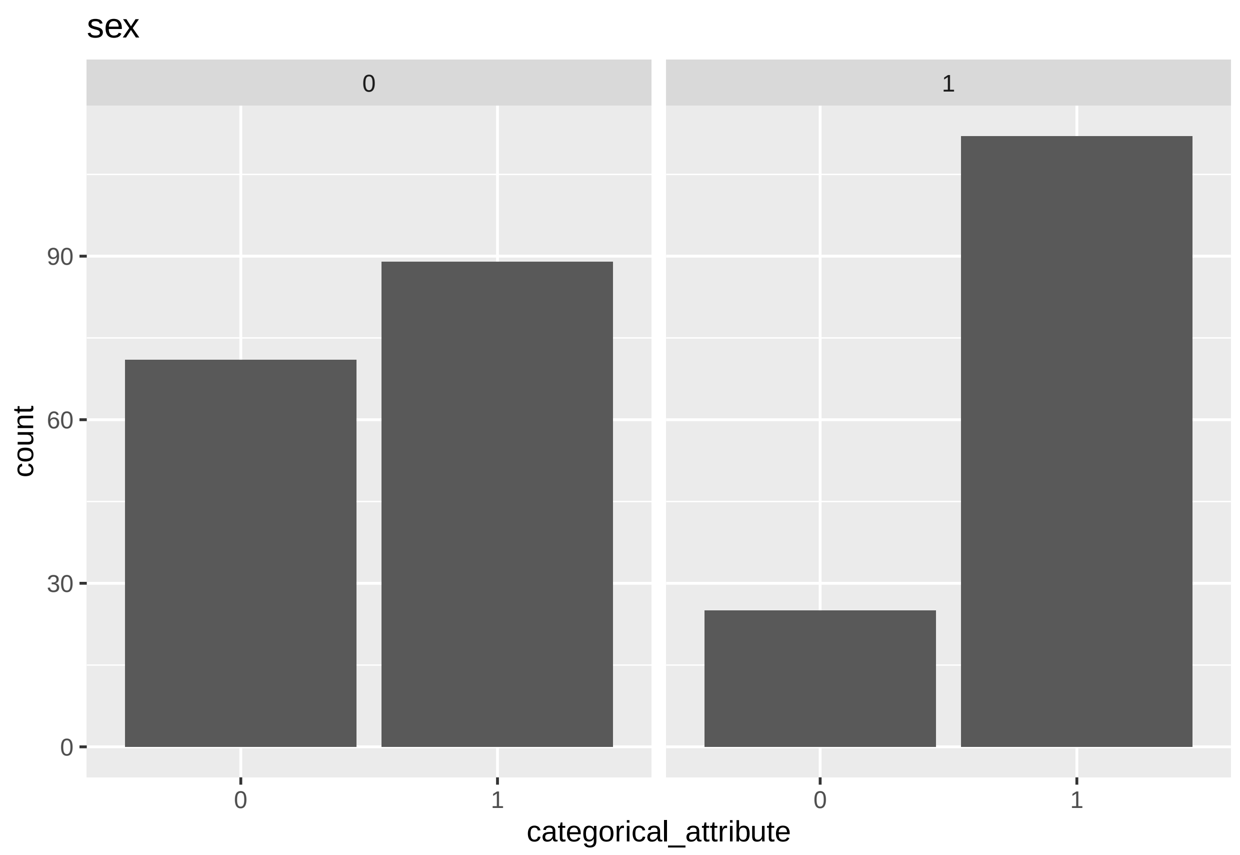
```
plot_histogram(cleveland_dataframe,cleveland_dataframe$oldpeak,"oldpeak")
```



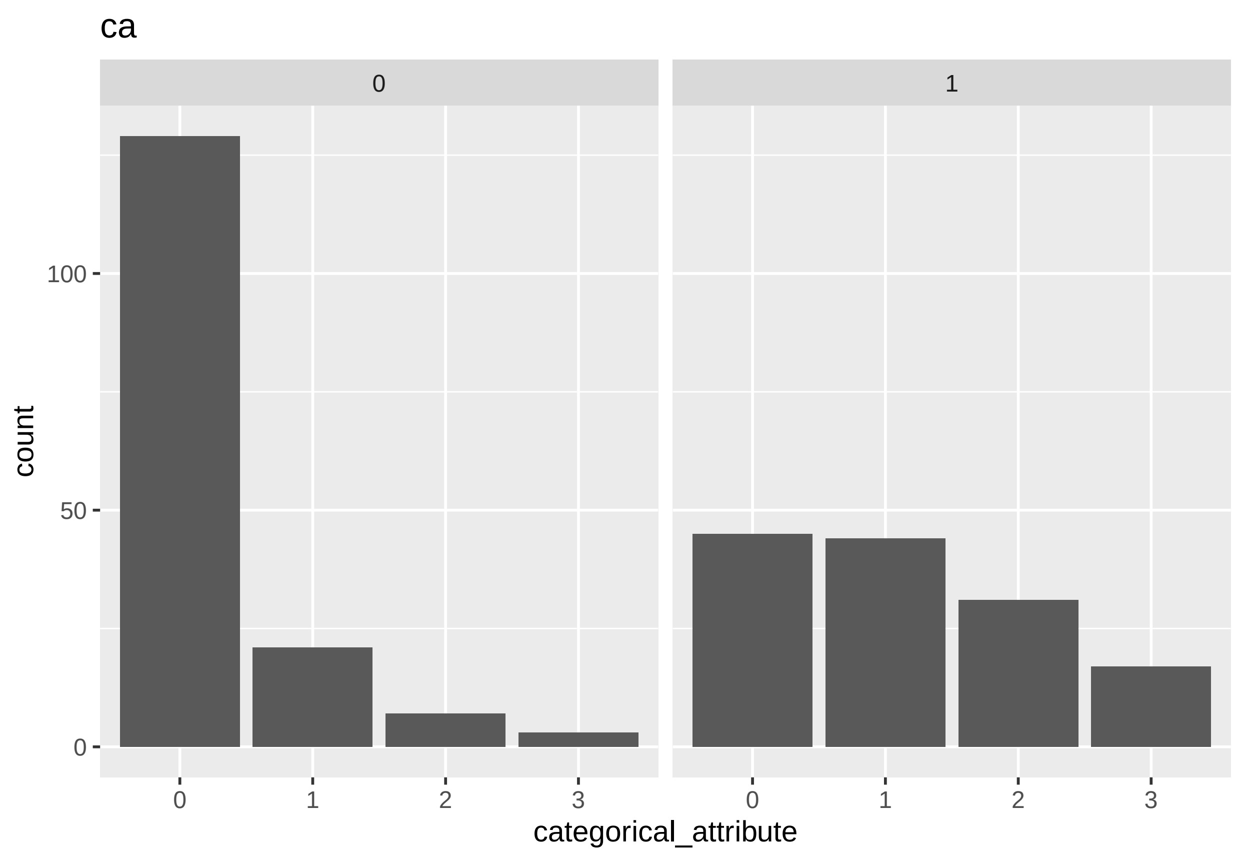
```
plot_histogram(cleveland_dataframe,cleveland_dataframe$age,"age")
```



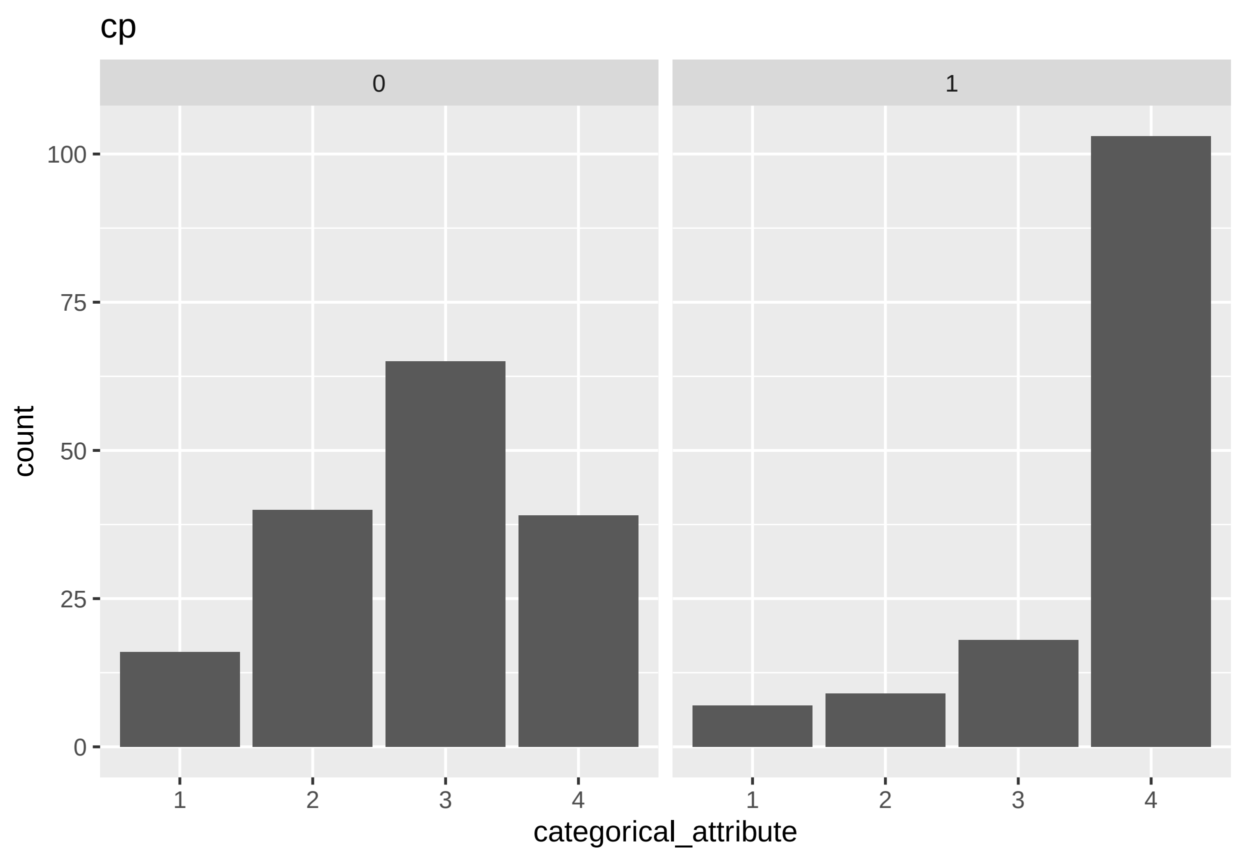
```
plot_bar(cleveland_dataframe,cleveland_dataframe$sex,"sex")
```



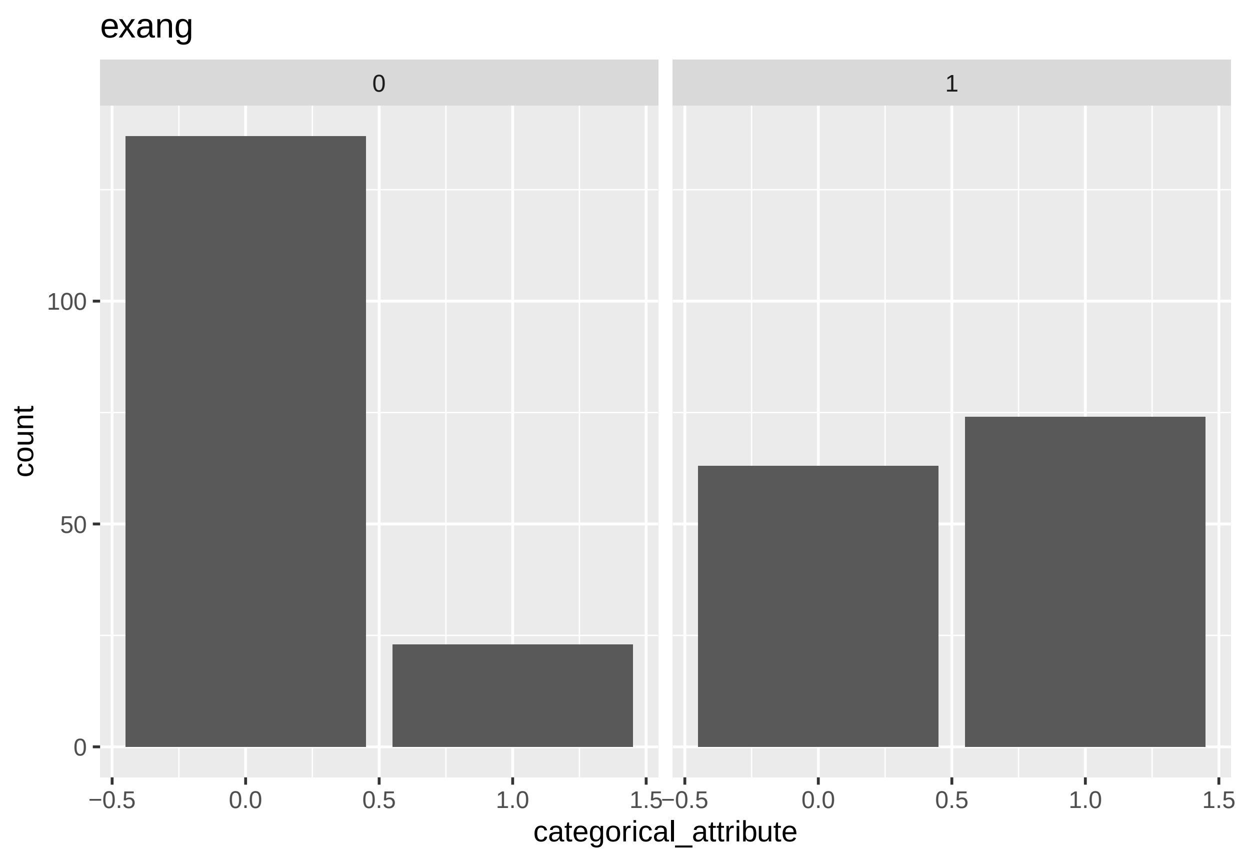
```
plot_bar(cleveland_dataframe,cleveland_dataframe$ca,"ca")
```



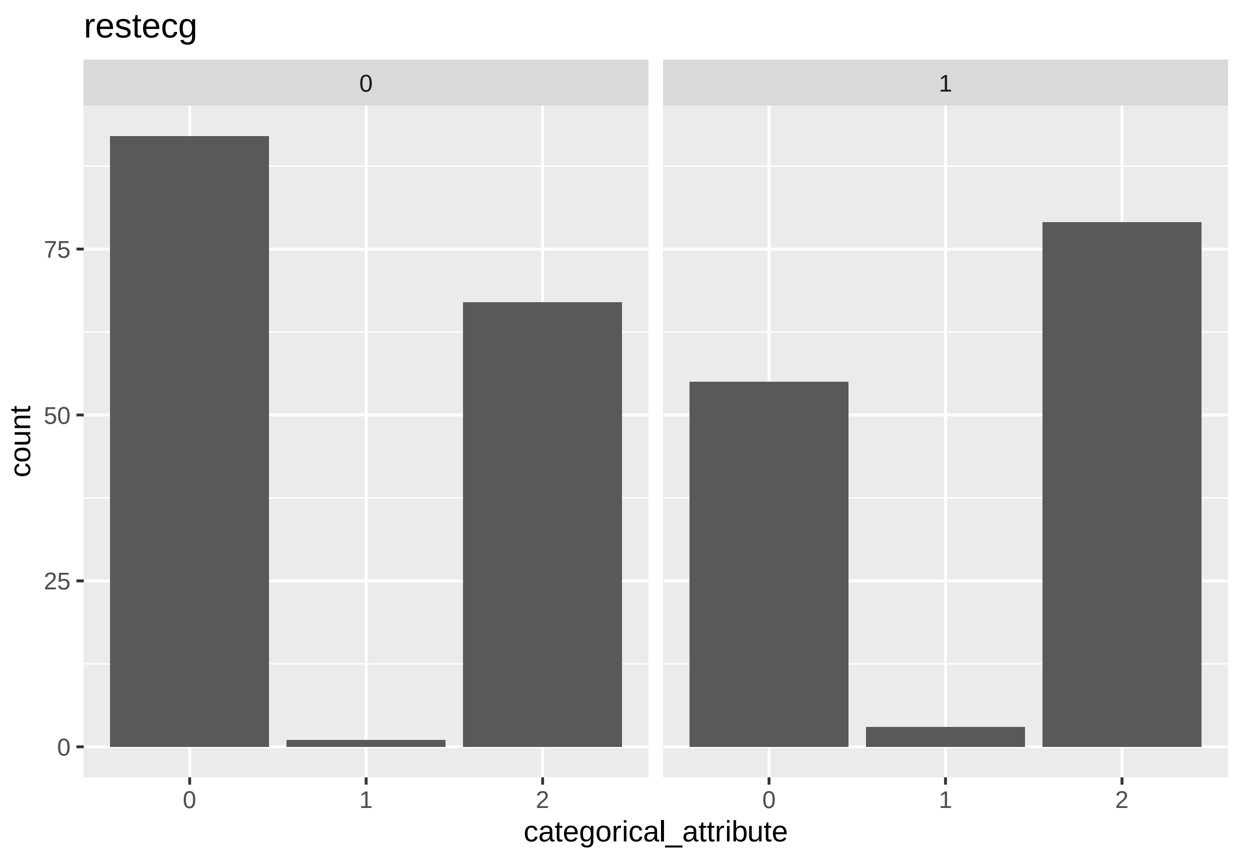
```r
plot_bar(cleveland_dataframe,cleveland_dataframe$cp,"cp")
```



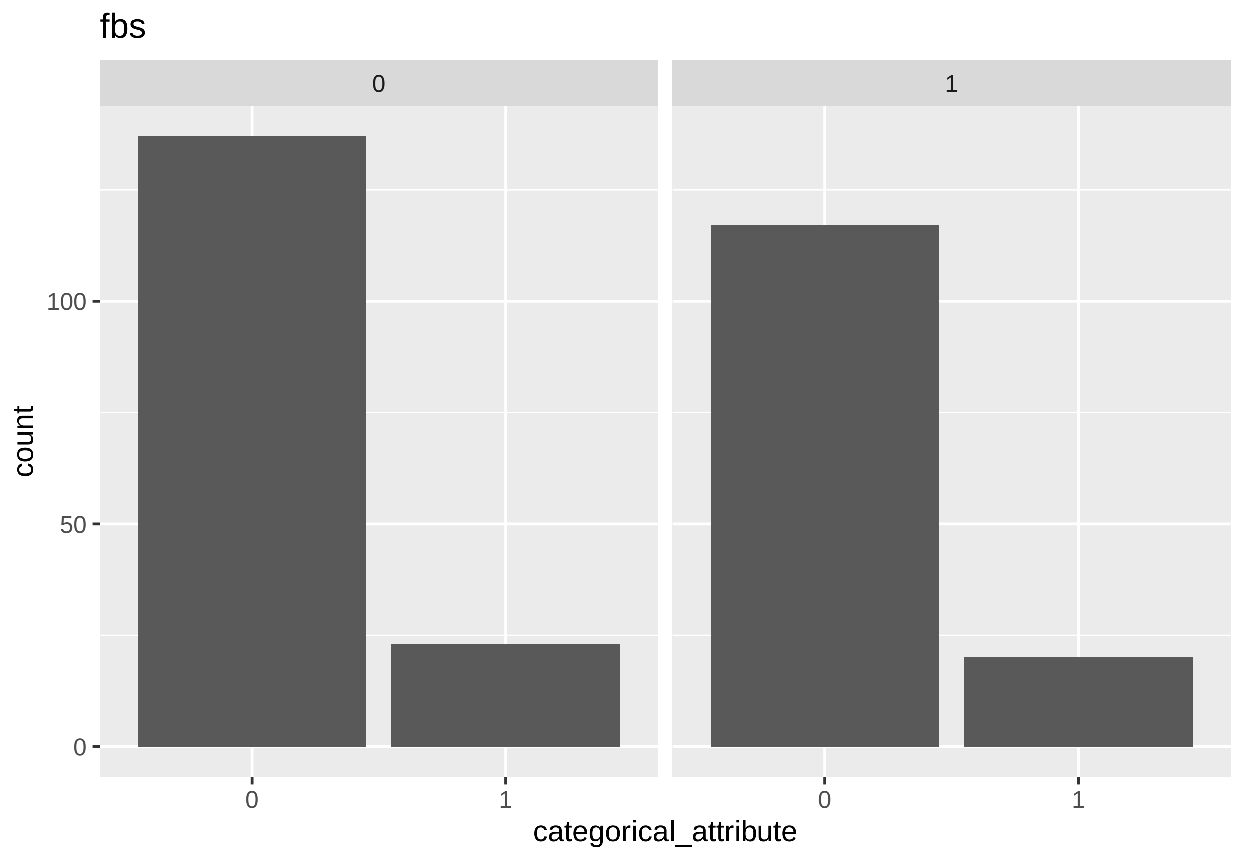
```
plot_bar(cleveland_dataframe,cleveland_dataframe$exang,"exang")
```



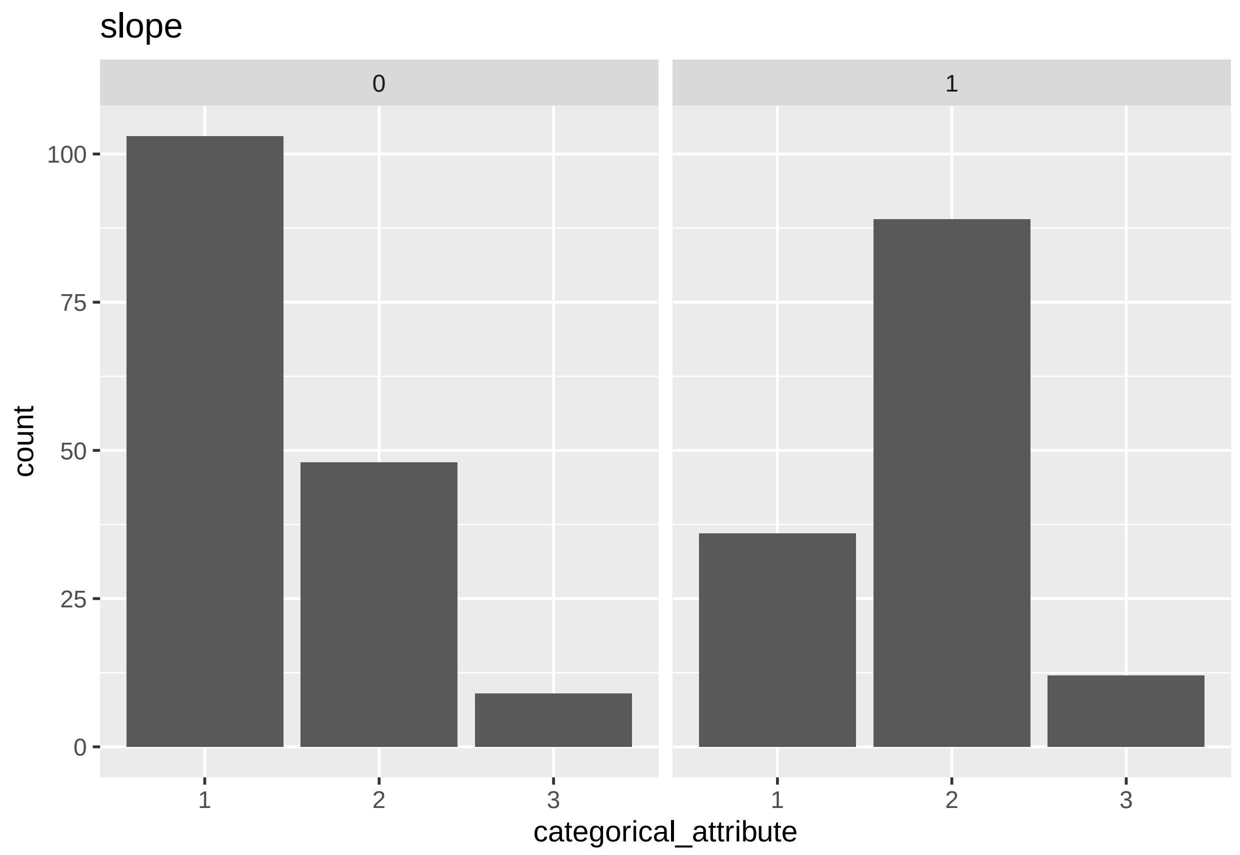
```r
plot_bar(cleveland_dataframe,cleveland_dataframe$restecg,"restecg")
```



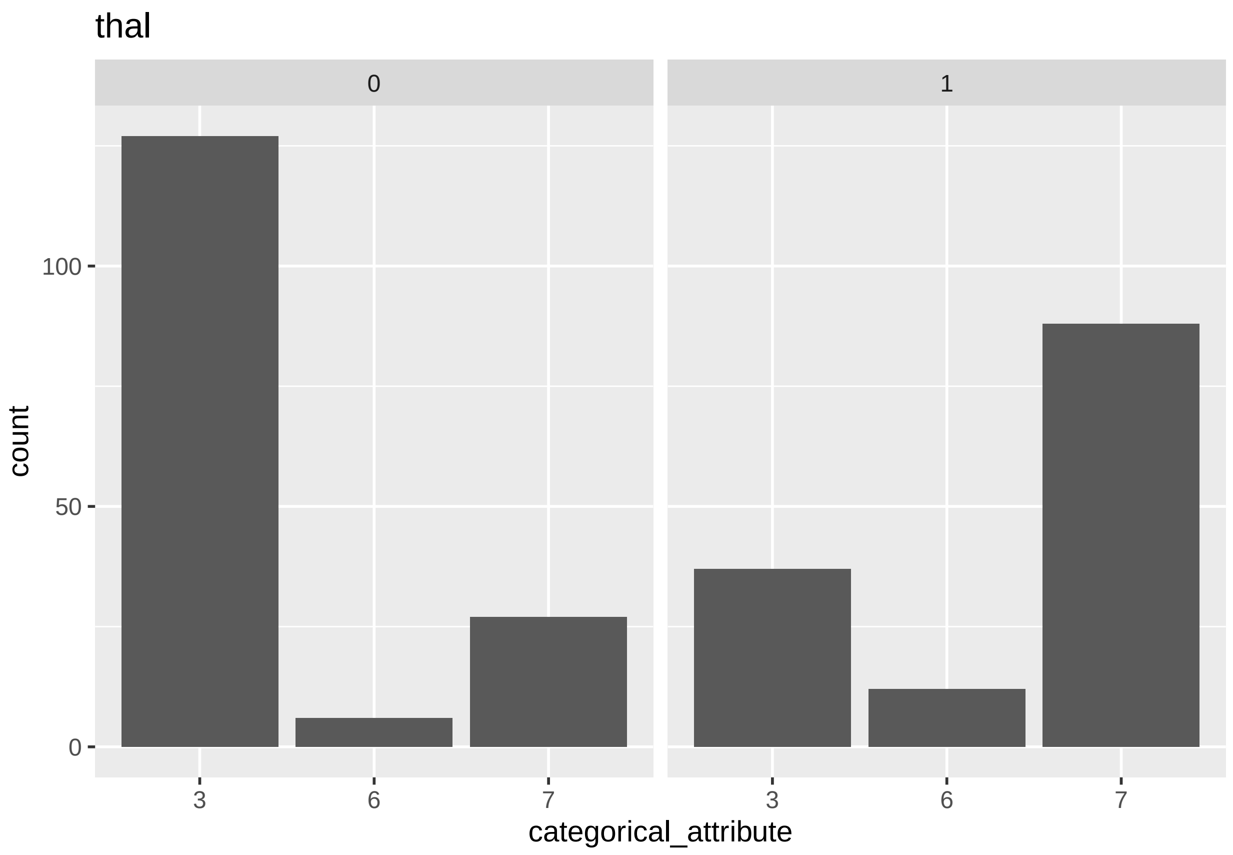
```
plot_bar(cleveland_dataframe,cleveland_dataframe$fbs,"fbs")
```



```
plot_bar(cleveland_dataframe,cleveland_dataframe$slope,"slope")
```



```r
plot_bar(cleveland_dataframe,cleveland_dataframe$thal,"thal")
```



Trestbps : It seems like the more num approch to 4, the more blood pressure increase. If the blood pressure is higher than 120 for certain people, there is higher chance to get a heart disease.

Chol : In this histogram, it seems like cholesterol affect a little bit on heart. Healthy person has their cholesterol around 200 mg/dl, whereas person with num 1,2,3 and 4 has their cholesterol around 250 mg/dl. But these values does not prove that cholesterol is a direct consequence of num.

Thalach : The maximum heart rate achieved is better on healthy person than person with heart disease. We could say that having a low heart rate can affect on heart.

Oldpeak : We can see that the distribution of values tends to shift on the right each time num approach to 4. This histogram shows us that the higher is oldpeak, the more we have chance to get a heart disease.

Age : Persons that have heart disease are mainly aged around 50 to 60 years old.

Sex : In Cleveland, people who are mainly touched by heart diseases are men.

ca : We can see that almost all healthy person has no vessels, but this does not mean so far that having 1 or 3 vessels can lead to heart disease. In fact, we can see that healthy person have also 1 to 3 vessels, so we can't say nothing.

cp : Most of the person with heart disease has the asymptomatic chest pain, which may be the main factor of heart disease.

exang : We can say that exercise induced angina is higher for those who have heart disease. Compared with healthy people, there is few people who's been affected to angina, but there is a huge proportion of people with no angina after exercise. We can say that exang is one of the factor that cause heart disease as well.

restcg : The proportion of normal rest electrocardiogram result seems to be obvious from healthy people, but we can also see that there is a huge proportion of healthy people that has a left ventricular hypertrophy.

Compared with those who have heart disease, we can't say nothing about the correlation between restcg and num.

fbs : Fast blood sugar seems not to be correlated to num. In fact, the proportion of fast blood sugar either > 120 or < 120 is the same for every num, so we can't say nothing too.

slope : Majority of healthy people have a upsloping peak exercise. On the opposite, majority people with heart disease has a flat peak exercise. So we could say that, having a flat slope peak exercise can induce to a heart disease.

thal : From the healthy person, majority of them have a normal thal. Compared to people with heart disease, most of them have a reversable thal which could be the cause of a heart disease.

To sum up, people from Cleveland are touched by heart disease around the age of 50-60. Most of them are men, and the cause of heart disease comes from several factor : a reversable thal, a flat slope, people that get angina and has an asymptomatic chest pain during exercise and a low heart rate achieved are the consequences of heart disease.

5.3 Modelling Approach

5.3.1 Training and Testing data for validation

Split the data into Training (70%) and Testing (30%) data. Percentage of heart disease or not must be same in training and testing (which is handled by the R-library used here).

```r
library(caret)
set.seed(10, sample.kind="Rounding")
cleveland_data_clean <- convert_magic(cleveland_data_clean,chclass)
inTrainRows <- createDataPartition(cleveland_data_clean$num,p=0.7,list=FALSE)
trainData <- cleveland_data_clean[inTrainRows,]
testData <-  cleveland_data_clean[-inTrainRows,]
nrow(trainData)/(nrow(testData)+nrow(trainData)) #checking whether really 70% -> OK
```

```
## [1] 0.7003367
```

5.3.2 Predict with different methods with different tuning parameters and compare best model of each method

Results are going to be stored in variable AUC. AUC is the area under the ROC which represents the proportion of positive data points that are correctly considered as positive and the proportion of negative data points that are mistakenly considered as positive. We also store Accuracy which is true positive and true negative divided by all results.

```r
AUC = list()
Accuracy = list()
```

5.3.2.1 Logistic regression

```r
set.seed(10, sample.kind="Rounding")
logRegModel <- train(num ~ ., data=trainData, method = 'glm', family = 'binomial')
logRegPrediction <- predict(logRegModel, testData)
logRegPredictionprob <- predict(logRegModel, testData, type='prob')[2]
logRegConfMat <- confusionMatrix(logRegPrediction, testData[,"num"])
```

```r
#ROC Curve
library(pROC)
AUC$logReg <- roc(as.numeric(testData$num),as.numeric(as.matrix((logRegPredictionprob))))$auc
Accuracy$logReg <- logRegConfMat$overall['Accuracy']

row.names <- names(Accuracy)
col.names <- c("AUC", "Accuracy")
cbind(as.data.frame(matrix(c(AUC,Accuracy),nrow = 1, ncol = 2,
                           dimnames = list(row.names, col.names))))
```

```
##              AUC  Accuracy
## logReg 0.9161585 0.8651685
```

The accuracy is 0.87 and AUC 0.92 which is already quite good.

5.3.2.2 Random Forest model without tuning (but checked a few number of trees)

```r
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.6.3

## 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
```

```r
set.seed(10, sample.kind="Rounding")
```

```
## Warning in set.seed(10, sample.kind = "Rounding"): non-uniform 'Rounding'
## sampler used
```

```r
RFModel <- randomForest(num ~ .,
                        data=trainData,
                        importance=TRUE,
                        ntree=2000)
#varImpPlot(RFModel)
RFPrediction <- predict(RFModel, testData)
RFPredictionprob = predict(RFModel,testData,type="prob")[, 2]

RFConfMat <- confusionMatrix(RFPrediction, testData[,"num"])

AUC$RF <- roc(as.numeric(testData$num),as.numeric(as.matrix((RFPredictionprob))))$auc
```

```
## Setting levels: control = 1, case = 2

## Setting direction: controls < cases
```

```
Accuracy$RF <- RFConfMat$overall['Accuracy']

row.names <- names(Accuracy)
col.names <- c("AUC", "Accuracy")
cbind(as.data.frame(matrix(c(AUC,Accuracy),nrow = 2, ncol = 2,
                           dimnames = list(row.names, col.names))))
```

```
##              AUC  Accuracy
## logReg 0.9161585 0.8651685
## RF     0.8932927 0.8202247
```

5.3.2.3 Boosted tree model with tuning (grid search)

Boosted tree model (gbm) with adjusting learning rate and and trees.

```
library(caret)
library(gbm)
set.seed(10, sample.kind="Rounding")
objControl <- trainControl(method='cv', number=10)
gbmGrid <-  expand.grid(interaction.depth =  c(1, 5, 9),
                        n.trees = (1:30)*50,
                        shrinkage = 0.1,
                        n.minobsinnode =10)
# run model
boostModel <- train(num ~ .,data=trainData, method='gbm',
                    trControl=objControl, tuneGrid = gbmGrid, verbose=F)

# See model output in Appendix to get an idea how it selects best model
boostPrediction <- predict(boostModel, testData)
boostPredictionprob <- predict(boostModel, testData, type='prob')[2]
boostConfMat <- confusionMatrix(boostPrediction, testData[,"num"])

#ROC Curve
AUC$boost <- roc(as.numeric(testData$num),as.numeric(as.matrix((boostPredictionprob))))$auc
Accuracy$boost <- boostConfMat$overall['Accuracy']

row.names <- names(Accuracy)
col.names <- c("AUC", "Accuracy")
cbind(as.data.frame(matrix(c(AUC,Accuracy),nrow = 3, ncol = 2,
                           dimnames = list(row.names, col.names))))
```

```
##              AUC  Accuracy
## logReg 0.9161585 0.8651685
## RF     0.8932927 0.8202247
## boost  0.9085366 0.8089888
```

6 Results

6.1 Comparison of AUC and Accuracy between models

```
row.names <- names(Accuracy)
col.names <- c("AUC", "Accuracy")
cbind(as.data.frame(matrix(c(AUC,Accuracy),nrow = 3, ncol = 2,
                           dimnames = list(row.names, col.names))))
```

```
##              AUC  Accuracy
## logReg 0.9161585 0.8651685
## RF     0.8932927 0.8202247
## boost  0.9085366 0.8089888
```

The best model is the relative simple logistic regression model with an Area under the ROC of 0.92. We can predict heart disease with an accuracy of 0.87. The Sensitivity is 0.90 and the Specificity 0.83.

6.2 Interpretation of logistic regression model and importance of variables from boosted tree

The coefficients of the 'best' model given AUC and Accuracy, the logistic regression model, are the following:

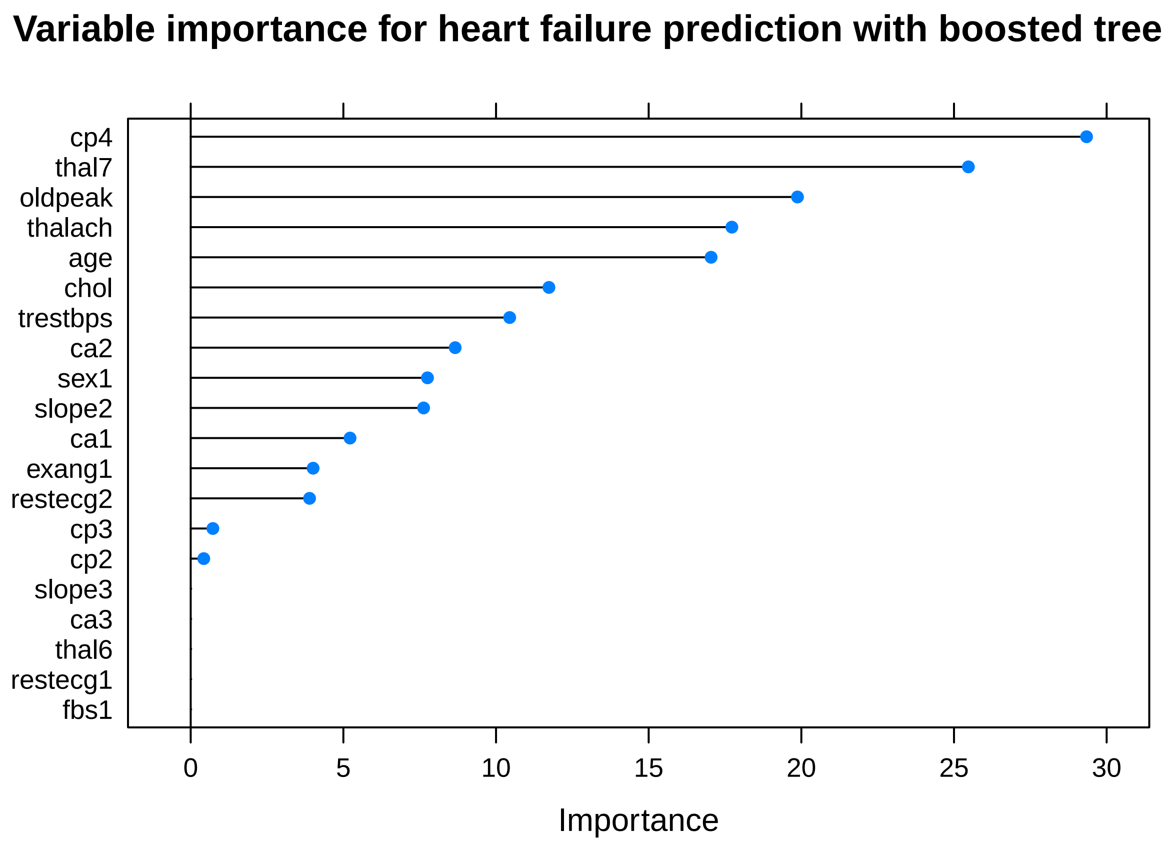
```
summary(logRegModel)$coeff
```

```
##                Estimate  Std. Error    z value     Pr(>|z|)
## (Intercept) -2.83507855 3.547759084 -0.7991181 0.4242219263
## age         -0.03180882 0.029637725 -1.0732545 0.2831569344
## sex1         1.85291093 0.711216844  2.6052686 0.0091802253
## cp2          0.44982427 1.005276129  0.4474634 0.6545405141
## cp3         -0.51437449 0.860554826 -0.5977243 0.5500239377
## cp4          1.64003231 0.868294403  1.8887975 0.0589189654
## trestbps     0.01714664 0.015671641  1.0941190 0.2739027719
## chol         0.00340254 0.004918579  0.6917730 0.4890799254
## fbs1        -0.24155269 0.804218007 -0.3003572 0.7639046843
## restecg2     0.25036985 0.492666693  0.5081932 0.6113178838
## thalach     -0.02492582 0.014363737 -1.7353301 0.0826823533
## exang1       0.52595947 0.555748185  0.9463989 0.3439451656
## oldpeak      0.20695564 0.284315718  0.7279078 0.4666700014
## slope2       1.72232845 0.616791265  2.7924008 0.0052318499
## slope3       1.03760679 1.069301753  0.9703592 0.3318674790
## ca1          2.56517830 0.696934041  3.6806615 0.0002326296
## ca2          3.94566322 0.994034355  3.9693429 0.0000720711
## ca3          2.16861195 1.010144229  2.1468340 0.0318065017
## thal6       -0.40962979 1.003136121 -0.4083492 0.6830173544
## thal7        1.58273584 0.549746570  2.8790281 0.0039890275
```

The interpretation of the coefficient for sex, for example, is: If all predictors are held at a fixed value, the odds of getting heart disease for males (males = 1) over the odds of getting heart disease for females is exp(1.85291093) = 6.4 i.e. the odds are 540% higher.

A direct comparison of the importance of each predictor is not possible for the logistic regression model. But this could be added in further analyses - comparing predictive ability of model by removing each variable seperately. Since the boosted tree model was only slightly lower, I here show the importance of the variables calculated by the boosted tree.

```r
boostImp =varImp(boostModel, scale = FALSE)
plot(boostImp,main = 'Variable importance for heart failure prediction with boosted tree')
```



7 Conclusion

14 predictor variables from the UCI heart disease dataset are used to predict the diagnosis of heart disease (angiographic disease status). The performances of 3 different machine learning algorithms - logistic regression, boosted trees, random forest and support vector machines - are compared . 70% of the data is hold out as a training data set that is not seen during the testing stage of the data. 30% of the data is hold out as a testing data set that is not seen during the training stage of the data. A comparison of the area under the ROC and the accuracy of the model predictions shows that logistic regression performs best (accuracy of 0.87). Tree-based methods with different tuning parameters performed slightly worse. Nevertheless, the boosted tree model was used to compare the importance of the different variables due to the easier procedure compared to logistic regression. The short analysis shows the predictive capability of machine learning algorithms for heart diseases.

8 Appendix

8.1 Confusion matrix output

8.1.1 Logistic Regression:

```
logRegConfMat
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 43  7
##          1  5 34
##
##                Accuracy : 0.8652
##                  95% CI : (0.7763, 0.9283)
##     No Information Rate : 0.5393
##     P-Value [Acc > NIR] : 5.93e-11
##
##                   Kappa : 0.7277
##
##  Mcnemar's Test P-Value : 0.7728
##
##             Sensitivity : 0.8958
##             Specificity : 0.8293
##          Pos Pred Value : 0.8600
##          Neg Pred Value : 0.8718
##              Prevalence : 0.5393
##          Detection Rate : 0.4831
##    Detection Prevalence : 0.5618
##       Balanced Accuracy : 0.8626
##
##        'Positive' Class : 0
##
```

8.1.2 Random Forest:

```
RFConfMat
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 40  8
##          1  8 33
##
##                Accuracy : 0.8202
##                  95% CI : (0.7245, 0.8936)
##     No Information Rate : 0.5393
##     P-Value [Acc > NIR] : 2.567e-08
##
##                   Kappa : 0.6382
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8333
##             Specificity : 0.8049
##          Pos Pred Value : 0.8333
##          Neg Pred Value : 0.8049
##              Prevalence : 0.5393
##          Detection Rate : 0.4494
##    Detection Prevalence : 0.5393
##       Balanced Accuracy : 0.8191
##
##        'Positive' Class : 0
##
```

8.1.3 Boosted Tree:

```
boostConfMat
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 41 10
##          1  7 31
##
##                Accuracy : 0.809
##                  95% CI : (0.7119, 0.8846)
##     No Information Rate : 0.5393
##     P-Value [Acc > NIR] : 9.657e-08
##
##                   Kappa : 0.6135
##
##  Mcnemar's Test P-Value : 0.6276
##
##             Sensitivity : 0.8542
##             Specificity : 0.7561
##          Pos Pred Value : 0.8039
##          Neg Pred Value : 0.8158
##              Prevalence : 0.5393
##          Detection Rate : 0.4607
##    Detection Prevalence : 0.5730
##       Balanced Accuracy : 0.8051
##
##        'Positive' Class : 0
##
```

8.2 Example of Model output for selection of tuning parameters

```
boostModel
```

```
## Stochastic Gradient Boosting
##
## 208 samples
##  13 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 188, 186, 187, 187, 187, 188, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                    50     0.8129870  0.6207890
##   1                   100     0.8120563  0.6167074
##   1                   150     0.8172727  0.6282051
##   1                   200     0.8220563  0.6377632
##   1                   250     0.8220563  0.6376036
##   1                   300     0.8079654  0.6106915
##   1                   350     0.7936797  0.5808689
##   1                   400     0.7886797  0.5706155
##   1                   450     0.7889177  0.5721431
##   1                   500     0.7793939  0.5520921
##   1                   550     0.7791558  0.5534702
##   1                   600     0.7843723  0.5640523
##   1                   650     0.7938961  0.5829701
##   1                   700     0.7843723  0.5636981
##   1                   750     0.7793723  0.5533491
##   1                   800     0.7746104  0.5435857
##   1                   850     0.7796104  0.5541114
##   1                   900     0.7841342  0.5630420
##   1                   950     0.7743939  0.5440667
##   1                  1000     0.7698485  0.5343609
##   1                  1050     0.7600866  0.5149222
##   1                  1100     0.7646320  0.5244335
##   1                  1150     0.7646320  0.5244305
##   1                  1200     0.7696320  0.5342138
##   1                  1250     0.7646320  0.5244305
##   1                  1300     0.7548701  0.5047973
##   1                  1350     0.7550866  0.5037071
##   1                  1400     0.7598485  0.5137396
##   1                  1450     0.7500866  0.4947429
##   1                  1500     0.7358009  0.4647726
##   5                    50     0.8227489  0.6420062
##   5                   100     0.8177489  0.6325921
##   5                   150     0.7991558  0.5943350
##   5                   200     0.7948485  0.5839983
##   5                   250     0.7993723  0.5924514
##   5                   300     0.7898485  0.5733669
##   5                   350     0.7848485  0.5627118
##   5                   400    0.7896320  0.5723531
##   5                   450    0.7846320  0.5617217
##   5                   500    0.7843939  0.5615304
##   5                   550    0.7796320  0.5518553
##   5                   600    0.7891558  0.5712038
##   5                   650    0.7843939  0.5630319
##   5                   700    0.7891558  0.5727053
##   5                   750    0.7941558  0.5820975
##   5                   800    0.7846320  0.5631486
##   5                   850    0.7896320  0.5727526
##   5                   900    0.7943939  0.5826657
##   5                   950    0.7946320  0.5833840
##   5                  1000    0.7993939  0.5932370
##   5                  1050    0.7993939  0.5932370
##   5                  1100    0.8041558  0.6029120
##   5                  1150    0.7946320  0.5833840
##   5                  1200    0.7946320  0.5833840
##   5                  1250    0.7896320  0.5733903
##   5                  1300    0.7946320  0.5833840
##   5                  1350    0.7946320  0.5833840
##   5                  1400    0.7946320  0.5833840
##   5                  1450    0.7946320  0.5833840
##   5                  1500    0.7946320  0.5833840
##   9                    50    0.8225325  0.6408087
##   9                   100    0.8129870  0.6220381
##   9                   150    0.7982251  0.5897878
##   9                   200    0.8077706  0.6094953
##   9                   250    0.8130087  0.6213385
##   9                   300    0.8082251  0.6111898
##   9                   350    0.8084416  0.6123587
##   9                   400    0.7986797  0.5924315
##   9                   450    0.7989177  0.5931126
##   9                   500    0.7984632  0.5921219
##   9                   550    0.7991558  0.5921011
##   9                   600    0.8039177  0.6038328
##   9                   650    0.8039177  0.6038328
##   9                   700    0.8039177  0.6038328
##   9                   750    0.8086797  0.6131286
##   9                   800    0.8036797  0.6031349
##   9                   850    0.8036797  0.6031349
##   9                   900    0.7989177  0.5924102
##   9                   950    0.7989177  0.5924102
##   9                  1000    0.7989177  0.5924102
##   9                  1050    0.7989177  0.5924102
##   9                  1100    0.8039177  0.6024039
##   9                  1150    0.8039177  0.6024039
##   9                  1200    0.8039177  0.6024039
##   9                  1250    0.8039177  0.6024039
##   9                  1300    0.8086797  0.6119521
##   9                  1350    0.8134416  0.6226938
##   9                  1400    0.8184416  0.6322977
##   9                  1450    0.8184416  0.6322977
##   9                  1500    0.8136797  0.6226243
##
```

```
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 50, interaction.depth =
##  5, shrinkage = 0.1 and n.minobsinnode = 10.
```

8.3 Environment

```
print("Operating System:")
```

```
## [1] "Operating System:"
```

```
version
```

```
##                _
## platform       x86_64-w64-mingw32
## arch           x86_64
## os             mingw32
## system         x86_64, mingw32
## status
## major          3
## minor          6.1
## year           2019
## month          07
## day            05
## svn rev        76782
## language       R
## version.string R version 3.6.1 (2019-07-05)
## nickname       Action of the Toes
```