Heart Failure Dataset Analysis

Stefano Andreoli, Federico Cesare Cattò, Andrea Matteo Re

2024-04-24

Introduction

For this project, we dealt with the Heart Failure Dataset (https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction).

Cardiovascular disease represents a significant public health challenge worldwide, with millions of people affected each year. The use of machine learning models can be a valuable resource in detecting risks at an early stage and implementing targeted preventive interventions. By analysing data that includes characteristics relevant to heart disease, predictive tools can be developed to help detect potential heart problems before they become severe, enabling early diagnosis and management.

Dataset and variables

The dataset was created from the combination of five datasets on twelve common characteristics. This combination makes the dataset the largest source of data on heart disease for research purposes to date.

It includes 918 observations with the following 12 characteristics:

- **Age**: age of the patient (years)
- **Sex**: gender of the patient
  - M: Male
  - F: Female
- **ChestPainType**: type of chest pain (the term 'Angina' comes from Latin and indicates the presence of a particular pain in the chest).
  - TA: typical angina
  - ATA: atypical angina (does not present with classical symptoms)
  - NAP: non-anginal pain
  - ASY: asymptomatic (absence of pain)
- **RestingBP**: resting blood pressure (mm/Hg)
- **Cholesterol**: serum cholesterol (mm/dl)
- **FastingBS**: fasting blood glucose
  - 1: if fasting blood glucose > 120 mg/dl
  - 0: otherwise
- **RestingECG**: resting electrocardiogram.
  - Normale: Normal
  - ST: with ST-T wave abnormality (T-wave inversions and/or ST-segment elevation or depression of > 0.05 mV)
  - LVH: showing left ventricular hypertrophy.

- **MaxHR**: maximum heart rate achieved (numerical value between 60 and 202).
- **ExerciseAngina**: exercise-induced angina.
  - Y:yes
  - N:no
- **Oldpeak**: ST-segment underslicing (a term that refers to the position of the ST-segment in the patient's ECG) under stress compared to a resting situation. (Fig.1)

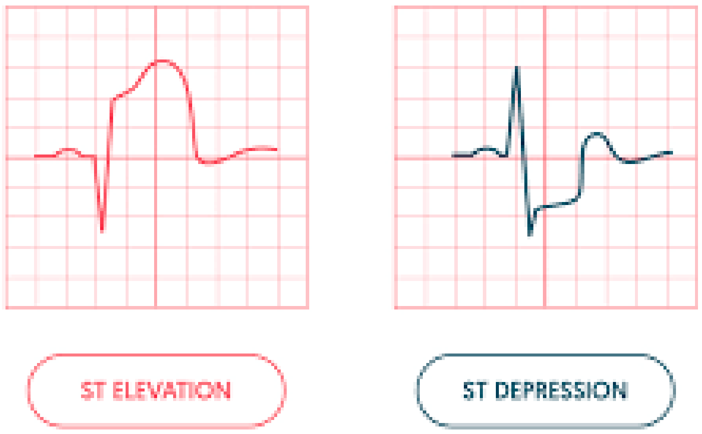


Figure 1: Fig.1: Examples of ST segment elevation or depression.

- **ST_Slope**: the slope of the ST segment at the moment of maximum effort. (Fig.2)
  - Up: growing
  - Flat: costant
  - Down: descending

- **HeartDisease**: target variable
  - 1: cardiac pathology
  - 0: normal

Data understanding & preparation

Inserting libraries and importing datasets.

```
library(MASS)
library(tidyverse)
library(ISLR)
library(tree)
library(ggcorrplot)
library(visdat)
```

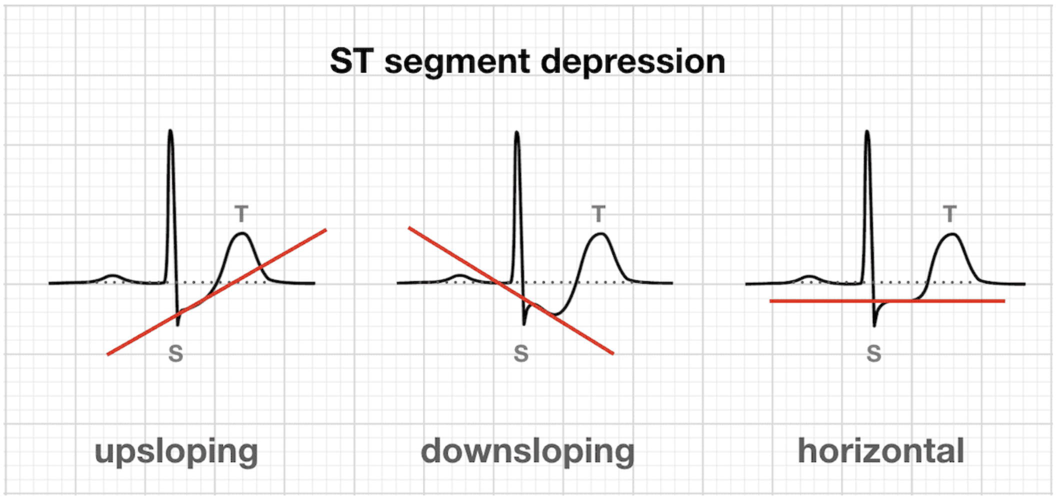


Figure 2: Fig.2: Types of ST segment slope

```
library(mice)
library(naniar)
library(UpSetR)
library(VIM)
library(GGally)
library(cowplot)
library(car)
library(gridExtra)
library(heplots)
library(mclust)
library(class)
library(caret)
library(ROCR)
select=dplyr::select
heart=read.csv("C:/Users/fccat/Documents/Projects/Heart_Failure_Analysis/heart.csv",stringsAsFactors = 
heart$FastingBS=as.factor(heart$FastingBS)
heart$HeartDisease=as.factor(heart$HeartDisease)
```

Missing values

We check for missing values in the dataset.

```
any_na(heart)
```

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

We proceed with a *summary* to check the ranges of the variables.

```
summary(heart)
```

```
##       Age          Sex      ChestPainType    RestingBP      Cholesterol
##  Min.   :28.00   F:193    ASY:496       Min.   :  0.0   Min.   :  0.0
##  1st Qu.:47.00   M:725   ATA:173       1st Qu.:120.0   1st Qu.:173.2  
##  Median :54.00           NAP:203       Median :130.0   Median :223.0  
##  Mean   :53.51           TA : 46       Mean   :132.4   Mean   :198.8  
##  3rd Qu.:60.00                         3rd Qu.:140.0   3rd Qu.:267.0  
##  Max.   :77.00                         Max.   :200.0   Max.   :603.0  
##  FastingBS  RestingECG      MaxHR       ExerciseAngina    Oldpeak       
##  0:704      LVH   :188   Min.   : 60.0   N:547          Min.   :-2.6000  
##  1:214      Normal:552   1st Qu.:120.0   Y:371          1st Qu.: 0.0000  
##             ST    :178   Median :138.0                  Median : 0.6000  
##                          Mean   :136.8                  Mean   : 0.8874  
##                          3rd Qu.:156.0                  3rd Qu.: 1.5000  
##                          Max.   :202.0                  Max.   : 6.2000  
##  ST_Slope   HeartDisease
##  Down: 63   0:410       
##  Flat:460   1:508       
##  Up  :395               
##                         
##                         
## 
```

The variables **RestingBP** and **Cholesterol** cannot take zero as a value, we note how this happens in our data. We deduce that they are missing values. We assign unique values to these to encode them.

```
heart$Cholesterol[heart$Cholesterol==0]=NA
heart$RestingBP[heart$RestingBP==0]=NA
summary(heart)
```

```
##       Age        Sex     ChestPainType   RestingBP      Cholesterol   
##  Min.   :28.00   F:193   ASY:496       Min.   : 80.0   Min.   : 85.0  
##  1st Qu.:47.00   M:725   ATA:173       1st Qu.:120.0   1st Qu.:207.2  
##  Median :54.00           NAP:203       Median :130.0   Median :237.0  
##  Mean   :53.51           TA : 46       Mean   :132.5   Mean   :244.6  
##  3rd Qu.:60.00                         3rd Qu.:140.0   3rd Qu.:275.0  
##  Max.   :77.00                         Max.   :200.0   Max.   :603.0  
##                                        NA's   :1       NA's   :172    
##  FastingBS  RestingECG      MaxHR       ExerciseAngina    Oldpeak       
##  0:704      LVH   :188   Min.   : 60.0   N:547          Min.   :-2.6000  
##  1:214      Normal:552   1st Qu.:120.0   Y:371          1st Qu.: 0.0000  
##             ST    :178   Median :138.0                  Median : 0.6000  
##                          Mean   :136.8                  Mean   : 0.8874  
##                          3rd Qu.:156.0                  3rd Qu.: 1.5000  
##                          Max.   :202.0                  Max.   : 6.2000  
##                                                                          
##  ST_Slope   HeartDisease
##  Down: 63   0:410       
##  Flat:460   1:508       
##  Up  :395               
##                         
##                         
##                         
## 
```

There are 172 missing values for the variable **Cholesterol**, and only one for **RestingBP**. We adopt a passive strategy for **RestingBP**, eliminating the one row with no value.

```
heart=heart%>%filter(!is.na(RestingBP))
```

As far as **Cholesterol** is concerned, we will adopt an active strategy.

Meanwhile, we deal with the splitting of the dataset into training, validation and test set.

```
set.seed(1)
test.index=sample(c(0:nrow(heart)),size = 0.2*nrow(heart),F)
test=heart[test.index,]
train_valid=anti_join(heart,test)
validation=train_valid%>%slice_sample(prop=0.2/0.8)
training=anti_join(train_valid,validation)
```

Balance check of the target class in the subsets created.

```
table(training$HeartDisease)/nrow(training)
```

```
##
##         0         1
## 0.4402174 0.5597826
```

```
table(validation$HeartDisease)/nrow(validation)
```

```
##
##         0         1
## 0.4699454 0.5300546
```

```
table(test$HeartDisease)/nrow(test)
```

```
##
##         0         1
## 0.4450549 0.5549451
```

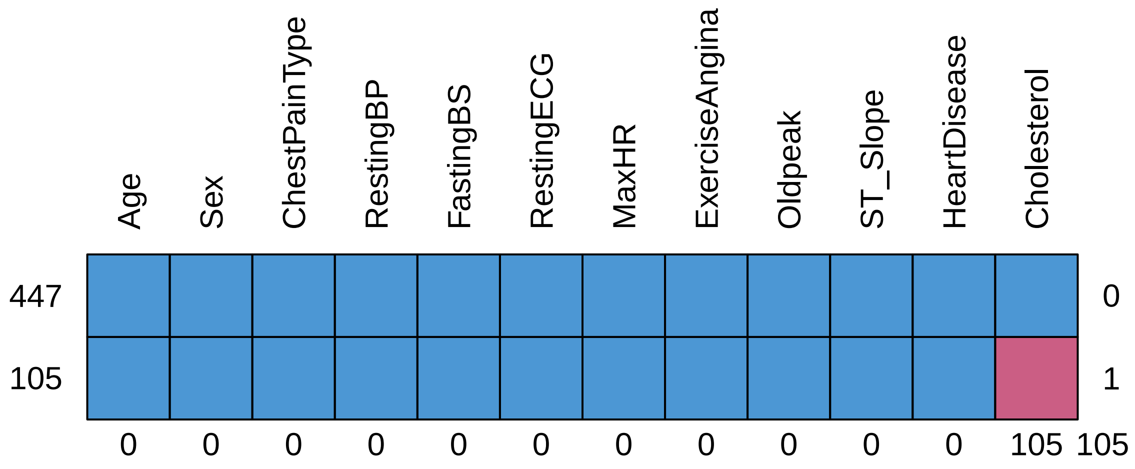
In addition to balancing between the various datasets, we also balance between the two classes of the target variable.

We create a function to calculate certain evaluation metrics such as *accuracy* and *sensitivity*, it will accept a confusion matrix as input.

```
metriche=function(tab){
  accuracy=sum(diag(tab))/sum(tab)
  sensitivity=tab[2,2]/sum(tab[,2])
  return(c("Accuracy"=accuracy,"Sensitivity"=sensitivity))
}
```

Evaluation of the type and distribution of missing values.

```
md.pattern(training, rotate.names=T)
```

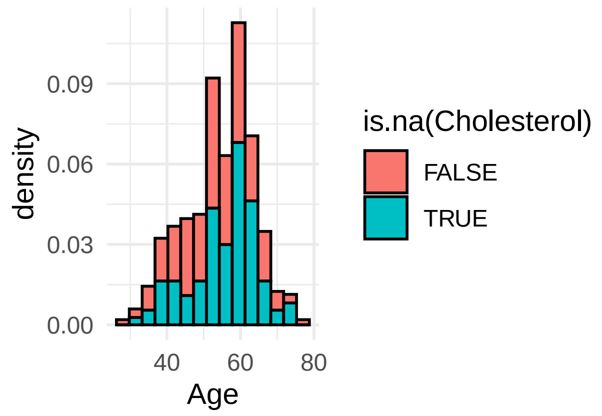


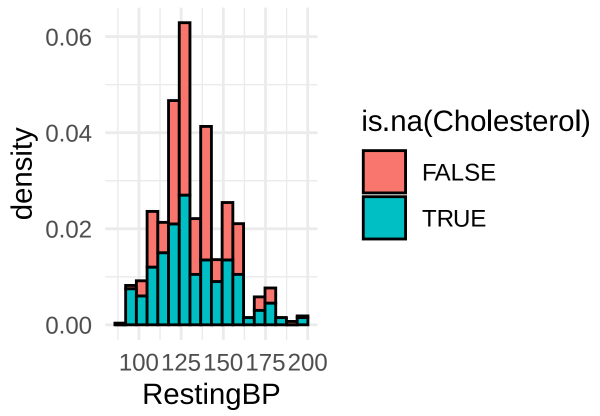
```
##     Age Sex ChestPainType RestingBP FastingBS RestingECG MaxHR ExerciseAngina
## 447   1   1             1         1         1          1     1              1
## 105   1   1             1         1         1          1     1              1
##       0   0             0         0         0          0     0              0
##     Oldpeak ST_Slope HeartDisease Cholesterol
## 447       1        1            1           1   0
## 105       1        1            1           0   1
##           0        0            0         105 105
```

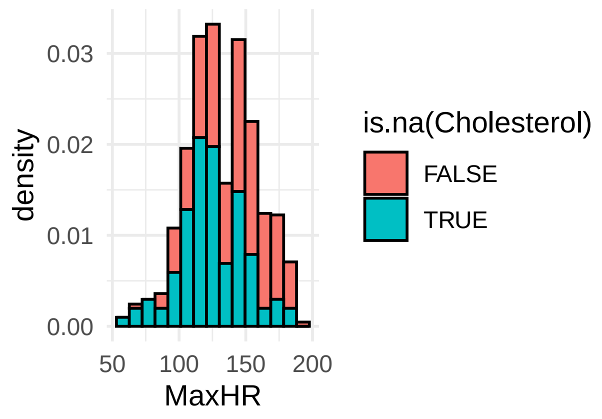
As we have seen, they are all concentrated in the variable **Cholesterol**.

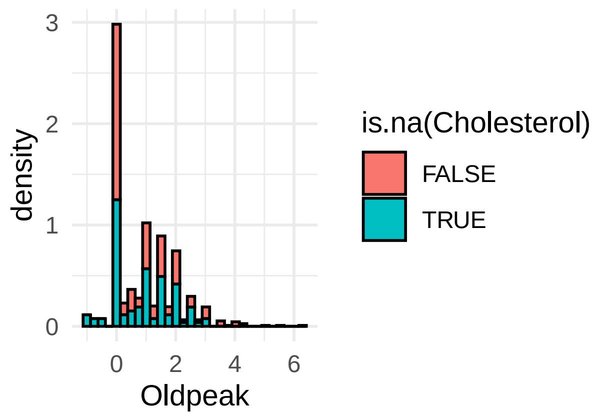
Let us now check the type of missing values (MAR,MCAR,NMAR).

```
g1=ggplot(training,aes(x=Age,y=after_stat(density)))+
  geom_histogram(aes(fill=is.na(Cholesterol)),col="black",bins=15)+
  theme_minimal()
g2=ggplot(training,aes(x=RestingBP,y=after_stat(density)))+
  geom_histogram(aes(fill=is.na(Cholesterol)),col="black",bins=18)+
  theme_minimal()
g3=ggplot(training,aes(x=MaxHR,y=after_stat(density)))+
  geom_histogram(aes(fill=is.na(Cholesterol)),col="black",bins=15)+
  theme_minimal()
g4=ggplot(training,aes(x=Oldpeak,y=after_stat(density)))+
  geom_histogram(aes(fill=is.na(Cholesterol)),col="black")+
  theme_minimal()
plot_grid(g1,g2,g3,g4)
```





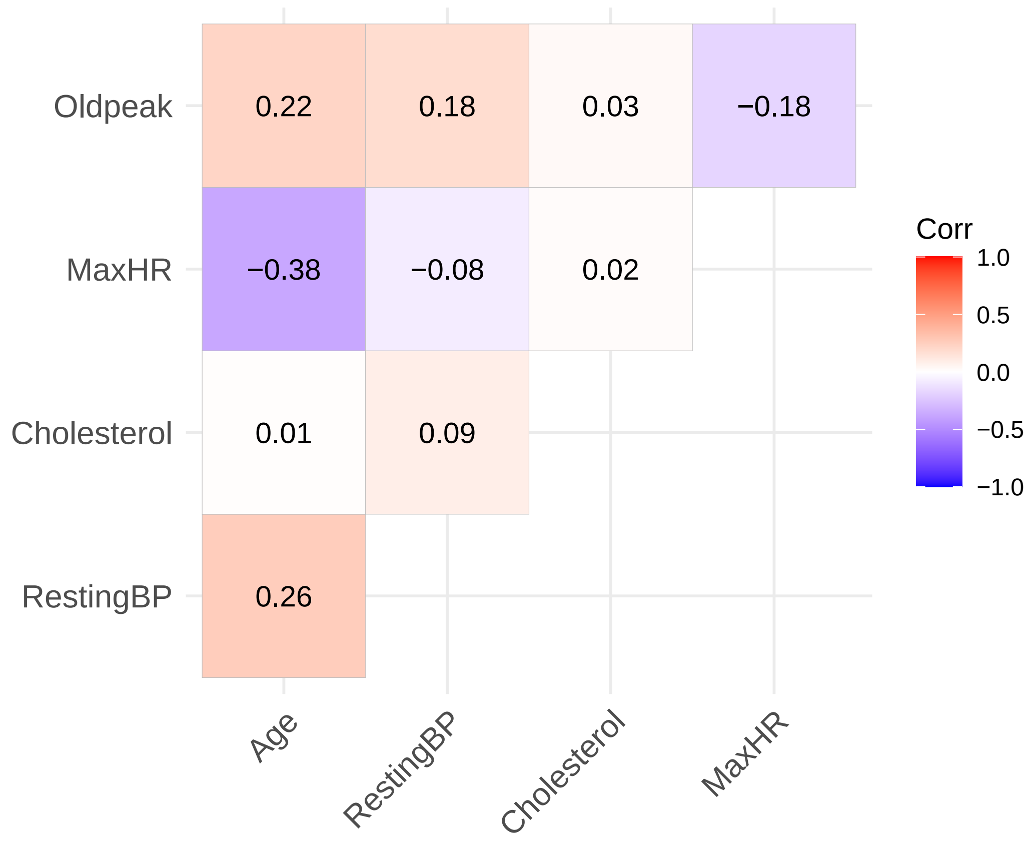




From the graphs above, we assume that the missing values are not NMAR.

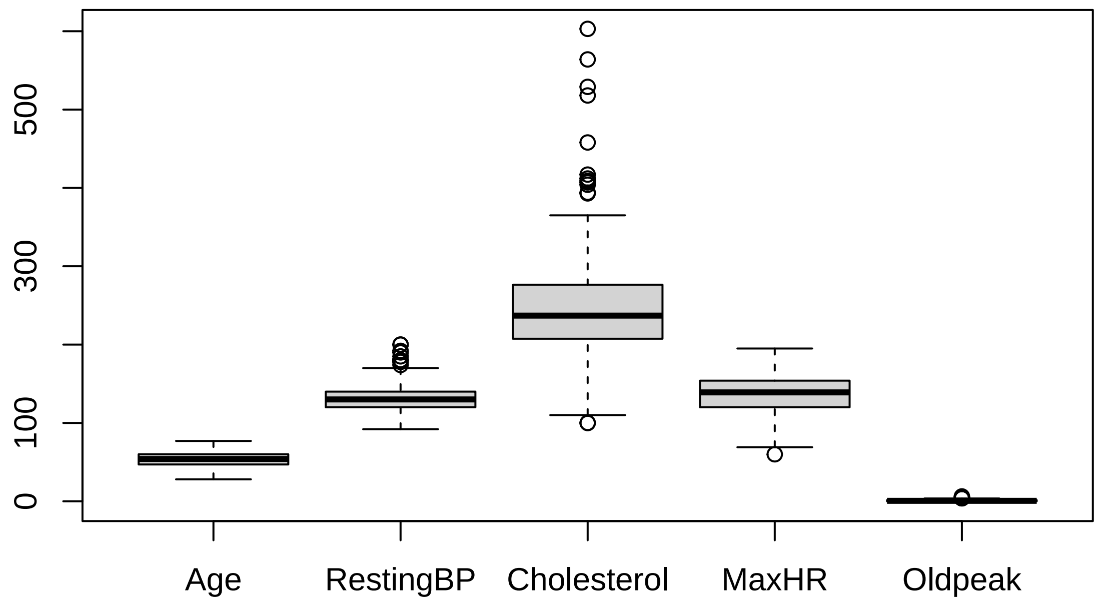
Let us examine the correlation between variables.

```r
ggcorrplot(cor(training%>%select(where(is.numeric)),use="pairwise.complete.obs"),show.diag = F,lab = T,
```



We visualise the distributions of the variables via boxplots, so that we can also observe whether outliers are present.

```
boxplot(training%>%select(where(is.numeric)))
```



The variables clearly assume values in different ranges.

Standardising

We have seen that the independent variables possess different ranges of variation, so let us perform a standardisation. As a position index we opt for the arithmetic mean while the mean square deviation will be our variability index; obviously both measures will be calculated on the training set and then used in the standardisation phase on the whole dataset.

```
means=colMeans(training%>%select(where(is.numeric)),na.rm=T)
variances=apply(training%>%select(where(is.numeric)),2,function(x){var(x,na.rm=T)})
training_non_stand=training
validation_non_stand=validation
test_non_stand=test
heart_non_stand=heart
training[,c("Age","RestingBP","Cholesterol","MaxHR","Oldpeak")]=scale(training%>%select(where(is.numeri
validation[,c("Age","RestingBP","Cholesterol","MaxHR","Oldpeak")]=scale(validation%>%select(where(is.nu
test[,c("Age","RestingBP","Cholesterol","MaxHR","Oldpeak")]=scale(test%>%select(where(is.numeric)),cent
heart[,c("Age","RestingBP","Cholesterol","MaxHR","Oldpeak")]=scale(heart%>%select(where(is.numeric)),ce
training_and_valid=rbind(training,validation)
```

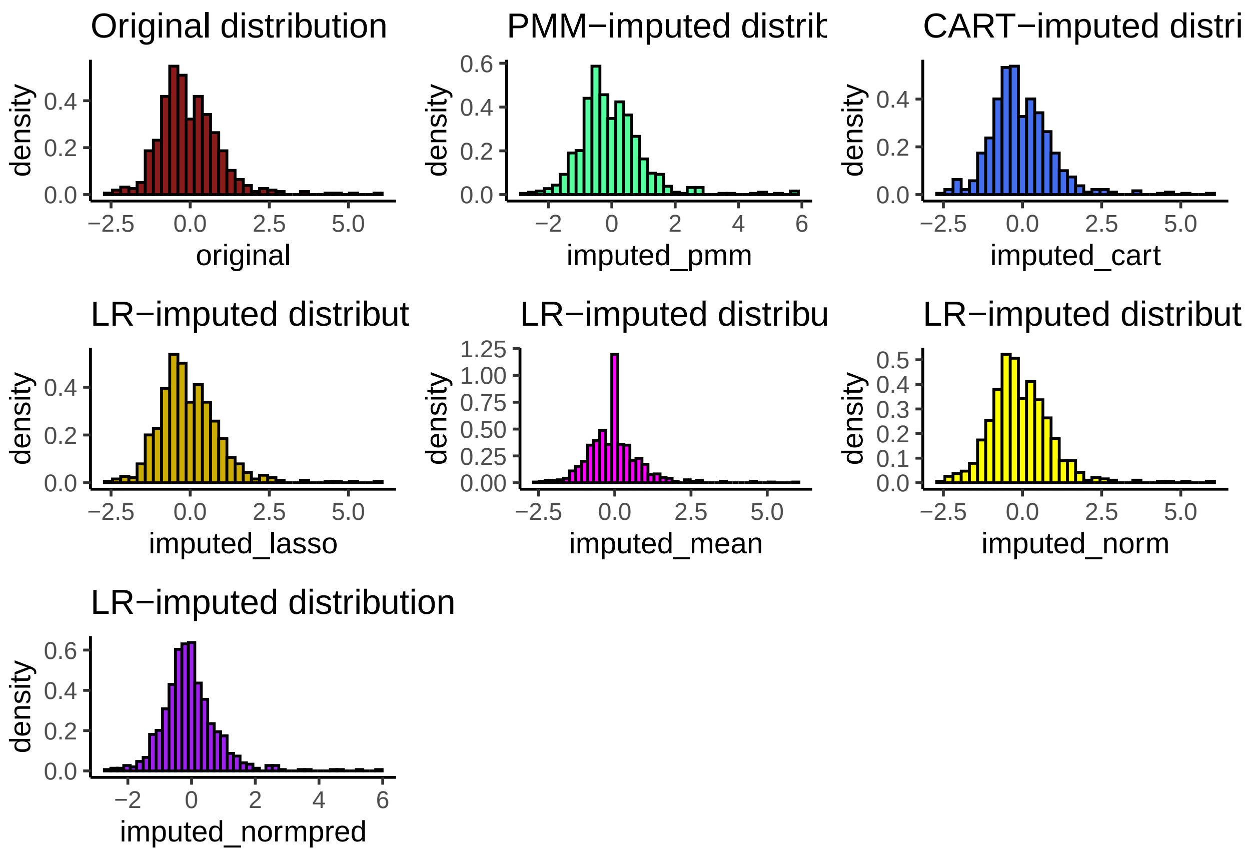
Imputation

We try out different imputation methods provided by the *mice* package, for the treatment of missing values in the variable **Cholesterol**.

```
matrix=matrix(1,nrow=12,ncol=12)
matrix[,12]=0
tempDatapmm=mice(training_and_valid,m=1,maxit=50,meth='pmm',seed=1,predictorMatrix = matrix,printFlag =
tempDatamean=mice(training_and_valid,m=1,maxit=50,meth='mean',seed=1,predictorMatrix = matrix,printFlag
tempDatanorm=mice(training_and_valid,m=1,maxit=50,meth='norm',seed=1,predictorMatrix = matrix,printFlag
tempDatanorm.nob=mice(training_and_valid,m=1,maxit=50,meth='norm.nob',seed=1,predictorMatrix = matrix,pr
tempDatanormpred=mice(training_and_valid,m=1,maxit=50,meth='norm.predict',seed=1,predictorMatrix = matri
tempDataLasso=mice(training_and_valid,m=1,maxit=50,meth='lasso.norm',seed=1,predictorMatrix = matrix,pri
tempDatacart=mice(training_and_valid,m=1,maxit=50,meth='cart',seed=1,predictorMatrix = matrix,printFlag
mice_imputed=data.frame(
  original = training_and_valid$Cholesterol,
  imputed_pmm = complete(tempDatapmm)$Cholesterol,
  imputed_cart = complete(tempDatacart)$Cholesterol,
  imputed_lasso = complete(tempDataLasso)$Cholesterol,
  imputed_mean = complete(tempDatamean)$Cholesterol,
  imputed_norm = complete(tempDatanorm)$Cholesterol,
  imputed_normpred = complete(tempDatanormpred)$Cholesterol
)
```

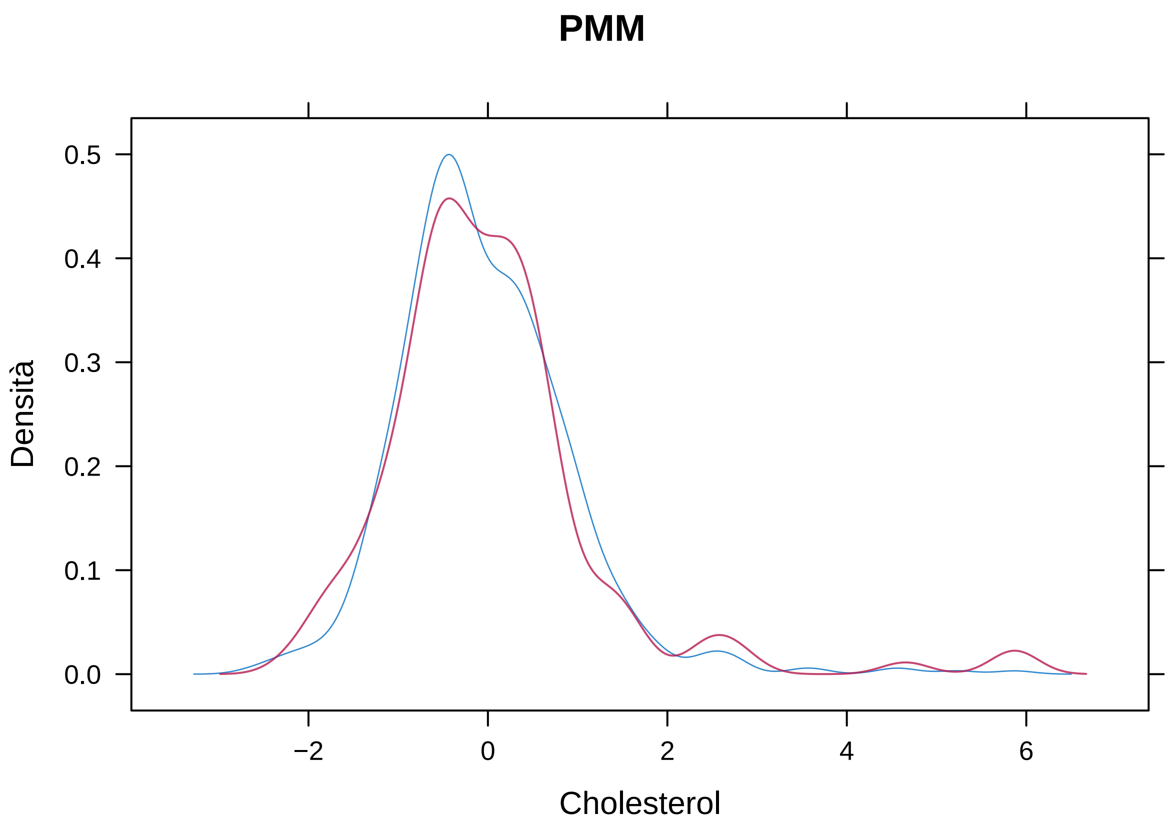
We will now have to choose the best method, selecting the one that best respects the original distribution of the variable **Cholesterol**. The comparison will be made using histograms.

```
bins_FD=round(apply(mice_imputed,2,function(x) (max(x,na.rm = T)-min(x,na.rm = T))/(2*IQR(x,na.rm = T))
mice_original=ggplot(mice_imputed, aes(x = original,y=after_stat(density))) +
  geom_histogram(fill = "firebrick4", color = "black", position = "identity",bins=bins_FD[1]) +
  ggtitle("Original distribution") +
  theme_classic()
mice_pmm= ggplot(mice_imputed, aes(x = imputed_pmm,y=after_stat(density))) +
  geom_histogram(fill = "seagreen1", color = "black", position = "identity",bins=bins_FD[2]) +
  ggtitle("PMM-imputed distribution") +
  theme_classic()
mice_cart = ggplot(mice_imputed, aes(x = imputed_cart,y=after_stat(density))) +
  geom_histogram(fill = "royalblue2", color = "black", position = "identity",bins=bins_FD[3]) +
  ggtitle("CART-imputed distribution") +
  theme_classic()
mice_lasso = ggplot(mice_imputed, aes(x = imputed_lasso,y=after_stat(density))) +
  geom_histogram(fill = "gold3", color = "black", position = "identity",bins=bins_FD[4]) +
  ggtitle("LR-imputed distribution") +
  theme_classic()
mice_mean = ggplot(mice_imputed, aes(x = imputed_mean,y=after_stat(density))) +
  geom_histogram(fill = "magenta", color = "black", position = "identity",bins=bins_FD[5]) +
  ggtitle("LR-imputed distribution") +
  theme_classic()
mice_norm = ggplot(mice_imputed, aes(x = imputed_norm,y=after_stat(density))) +
  geom_histogram(fill = "yellow", color = "black", position = "identity",bins=bins_FD[6]) +
  ggtitle("LR-imputed distribution") +
  theme_classic()
mice_normpred = ggplot(mice_imputed, aes(x = imputed_normpred,y=after_stat(density))) +
  geom_histogram(fill = "purple", color = "black", position = "identity",bins=bins_FD[7]) +
  ggtitle("LR-imputed distribution") +
  theme_classic()
plot_grid(mice_original,mice_pmm,mice_cart,mice_lasso,mice_mean,mice_norm,mice_normpred)
```

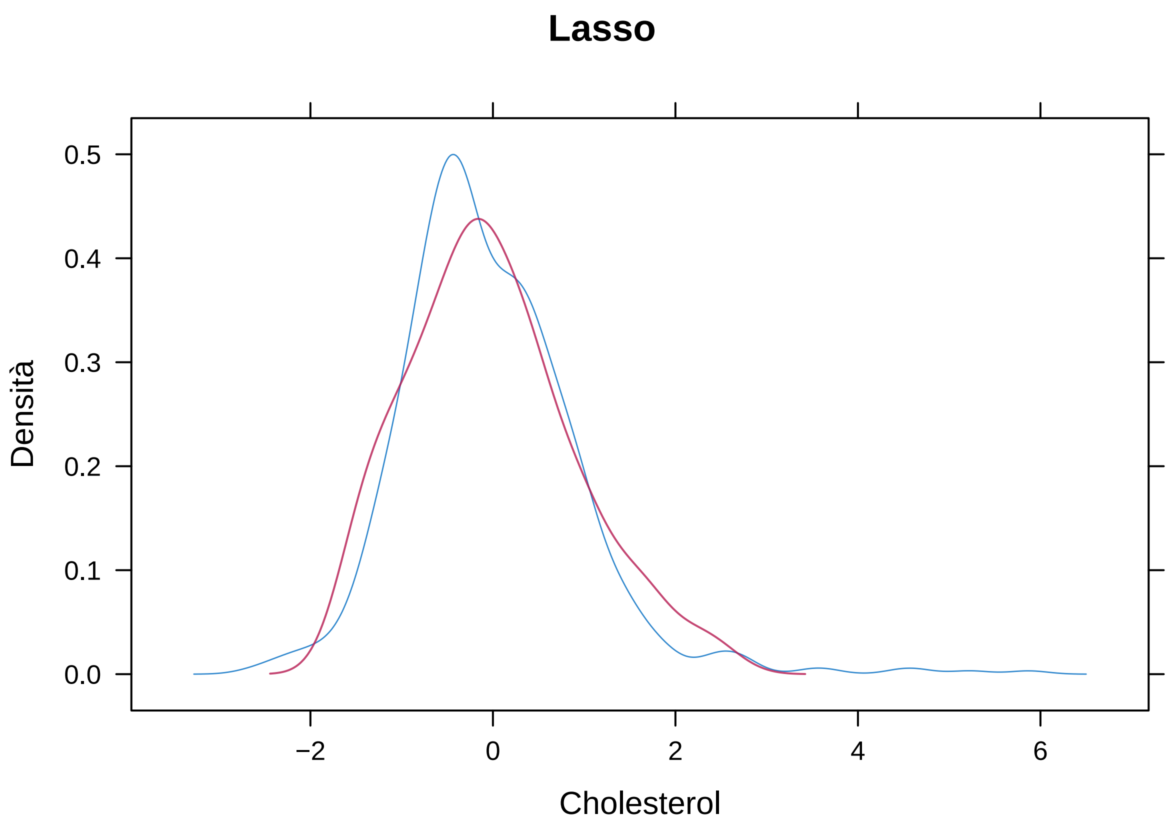


As there is no method that is clearly superior to the others, we choose the three best methods and make a further comparison via the *densityplot*.

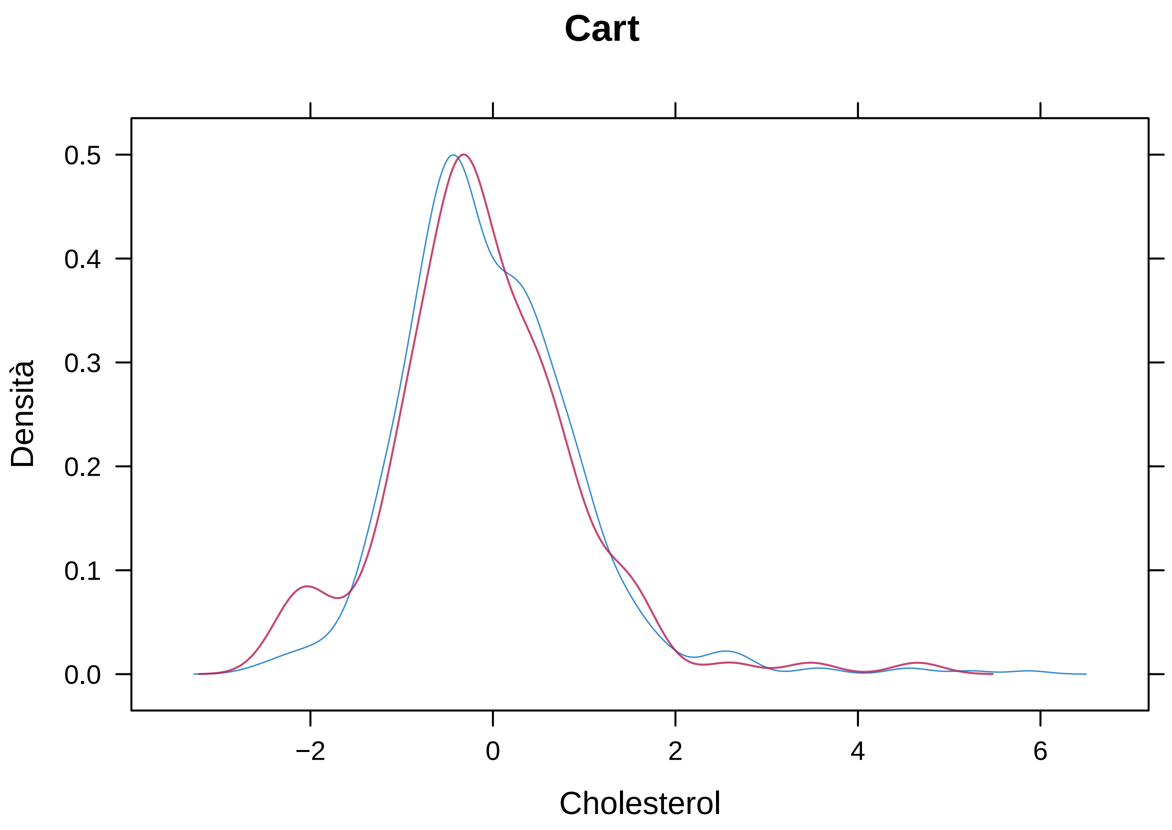
```
dens_pmm=densityplot(tempDatapmm,thicker = 0.7,main="PMM")
dens_Lasso=densityplot(tempDataLasso,thicker = 0.7,main="Lasso")
dens_cart=densityplot(tempDatacart,thicker = 0.7,main="Cart")
plot(dens_pmm)
```



```r
plot(dens_Lasso)
```



```r
plot(dens_cart)
```



The most reasonable method seems to be the *cart*, let us move on to imputation on the whole dataset.

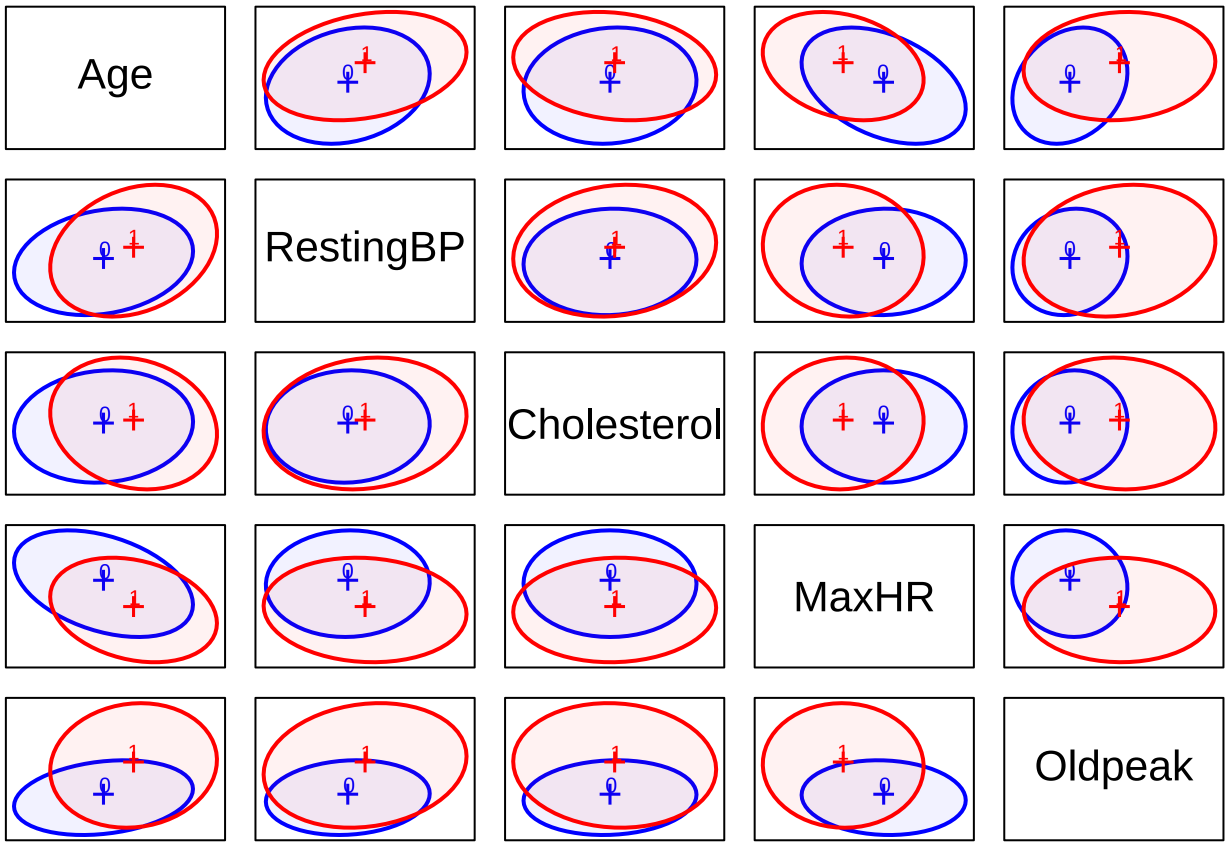
```
training_original=training
validation_original=validation
heart_imputed=mice(heart,predictorMatrix = matrix,meth="cart",m=1,printFlag = F,maxit = 50,ignore=c(1:n
test$Cholesterol=complete(heart_imputed)[test.index,]
training=inner_join(training_original,complete(heart_imputed),join_by(Age,Sex,ChestPainType,RestingBP,F
  mutate(Cholesterol=Cholesterol.y)%>%
  select(!contains("."))
validation=inner_join(validation_original,complete(heart_imputed),join_by(Age,Sex,ChestPainType,Resting
  mutate(Cholesterol=Cholesterol.y)%>%
  select(!contains("."))
```

Verifying assumptions

In this section we will test the applicability of techniques such as linear discriminant analysis (*LDA*) and quadratic discriminant analysis (*QDA*). These techniques assume a normal distribution of the covariates conditional on the mode of the target.

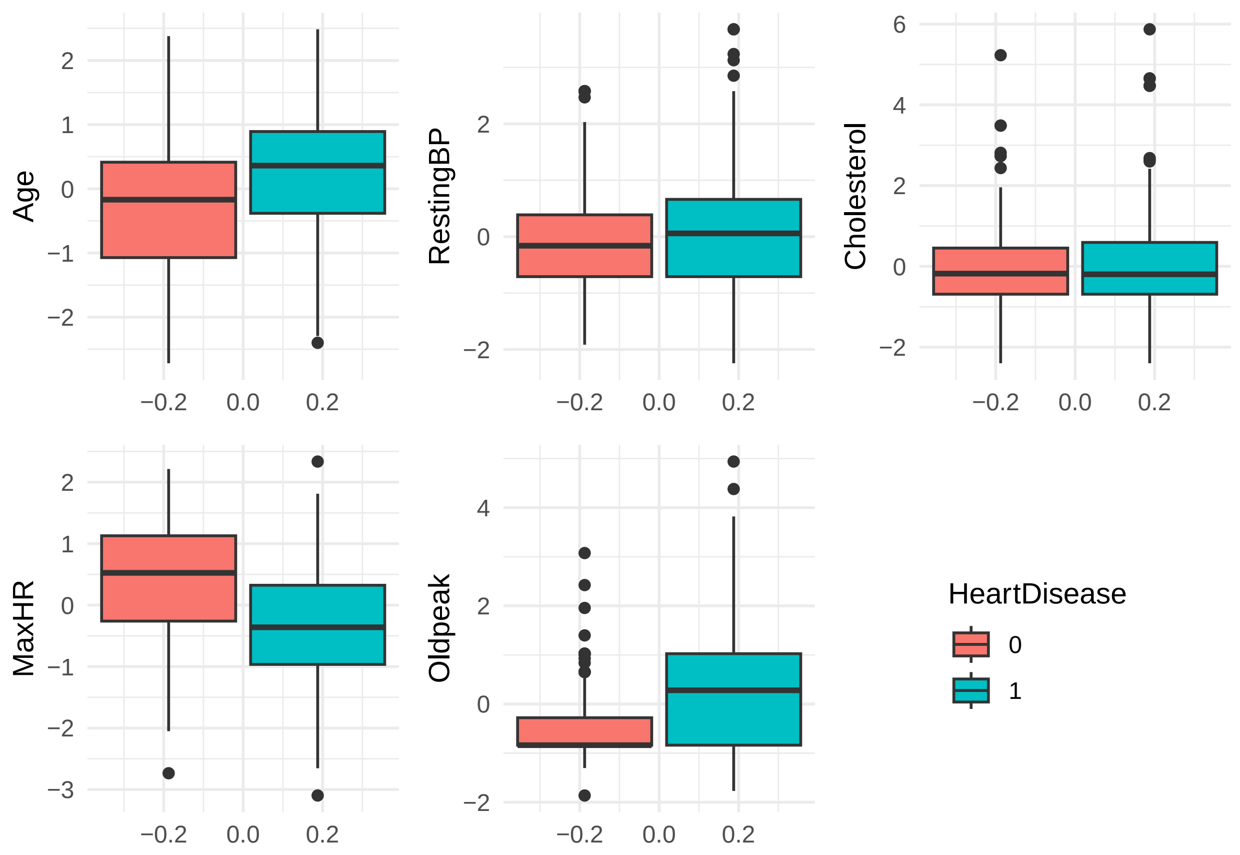
We check graphically whether the variables have covariance in common.

```
covEllipses(training%>%select(where(is.numeric)),
            factor(training$HeartDisease),
            fill = TRUE,
            pooled = FALSE,
            col = c("blue", "red"),
            variables=c("Age","RestingBP","Cholesterol","MaxHR","Oldpeak"),
            fill.alpha = 0.05)
```



An additional indication is provided by the conditional boxplots.

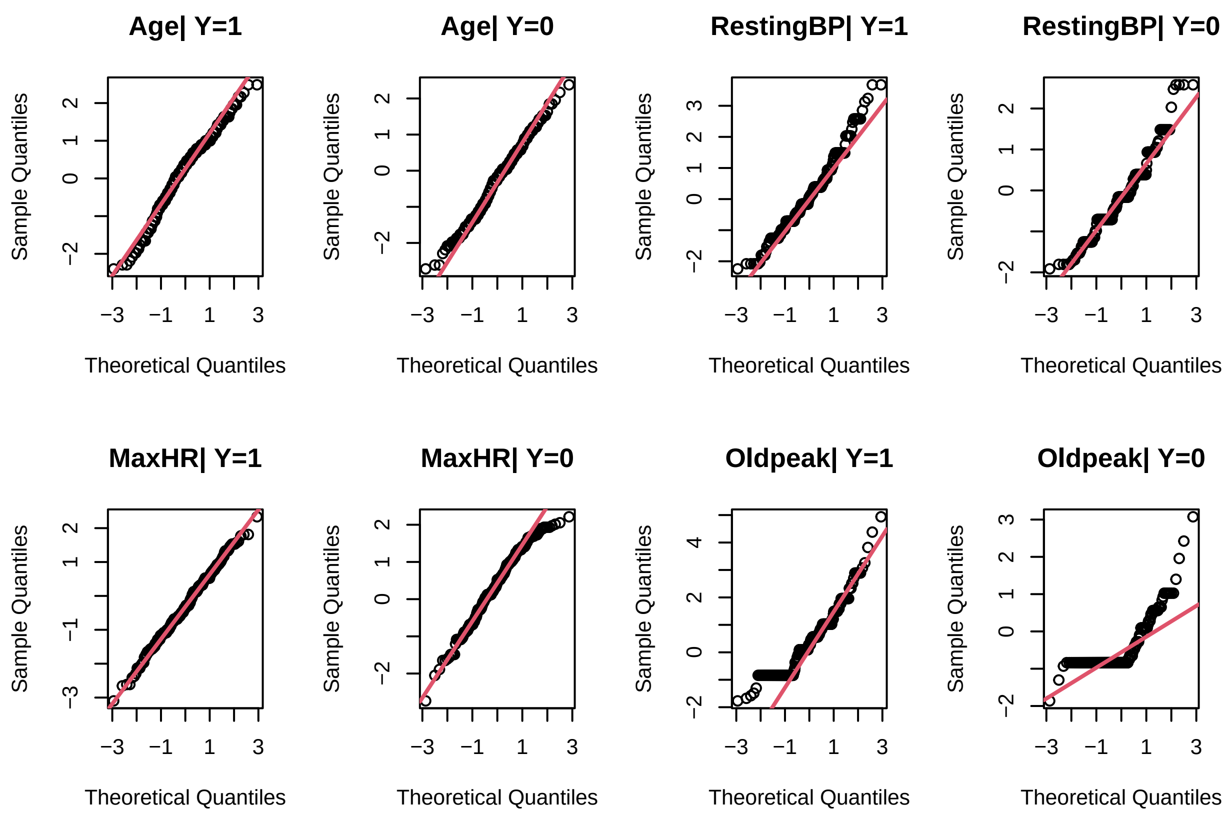
```
box1=ggplot(training,aes(y=Age,fill=HeartDisease))+
  geom_boxplot(show.legend = F)+
  theme_minimal()
box2=ggplot(training,aes(y=RestingBP,fill=HeartDisease))+
  geom_boxplot(show.legend = F)+
  theme_minimal()
box3=ggplot(training,aes(y=Cholesterol,fill=HeartDisease))+
  geom_boxplot(show.legend = F)+
  theme_minimal()
box4=ggplot(training,aes(y=MaxHR,fill=HeartDisease))+
  geom_boxplot(show.legend = F)+
  theme_minimal()
box5=ggplot(training,aes(y=Oldpeak,fill=HeartDisease))+
  geom_boxplot(show.legend = F)+
  theme_minimal()
box_legend=get_legend(ggplot(training,aes(y=Age,fill=HeartDisease))+
                        geom_boxplot(show.legend = T)+
                        theme_minimal())
plot_grid(box1,box2,box3,box4,box5,box_legend,ncol=3)
```



The information given by the boxplots is rather meagre, so let us move on to more accurate graphs.

In order to check the conditional normality of variables, we first use *qqplots*..

```
par(mfrow=c(2,4))
qqnorm(training$Age[training$HeartDisease==1],main = "Age| Y=1");qqline(training$Age[training$HeartDisea
qqnorm(training$Age[training$HeartDisease==0],main = "Age| Y=0");qqline(training$Age[training$HeartDisea
qqnorm(training$RestingBP[training$HeartDisease==1],main = "RestingBP| Y=1");qqline(training$RestingBP[t
qqnorm(training$RestingBP[training$HeartDisease==0],main = "RestingBP| Y=0");qqline(training$RestingBP[t
qqnorm(training$MaxHR[training$HeartDisease==1],main = "MaxHR| Y=1");qqline(training$MaxHR[training$Hear
qqnorm(training$MaxHR[training$HeartDisease==0],main = "MaxHR| Y=0");qqline(training$MaxHR[training$Hear
qqnorm(training$Oldpeak[training$HeartDisease==1],main = "Oldpeak| Y=1");qqline(training$Oldpeak[trainin
qqnorm(training$Oldpeak[training$HeartDisease==0],main = "Oldpeak| Y=0");qqline(training$Oldpeak[trainin
```



```r
par(mfrow=c(1,1))
```

From the qqplots we can see that some variables seem to distribute normally, while others are far from normal. For objective results we make use of some tests, such as the *Kolmogorov-Smirnov* and the *Shapiro* test.

```r
apply(training%>%filter(HeartDisease==1)%>%select(where(is.numeric)),2,function(x) ks.test(x,y="dnorm")
```

```
##           Age    RestingBP        MaxHR      Oldpeak  Cholesterol
## 3.791737e-259 1.432838e-268 5.441523e-255 8.093286e-269 8.073228e-269
```

```r
apply(training%>%filter(HeartDisease==1)%>%select(where(is.numeric)),2,function(x) shapiro.test(x)$p.va
```

```
##          Age    RestingBP        MaxHR      Oldpeak  Cholesterol
## 3.481059e-04 5.919100e-05 7.655581e-01 1.145152e-10 1.419618e-13
```

```r
apply(training%>%filter(HeartDisease==0)%>%select(where(is.numeric)),2,function(x) ks.test(x,y="dnorm")
```

```
##           Age    RestingBP        MaxHR      Oldpeak  Cholesterol
## 1.134255e-201 1.716959e-205 3.062842e-197 5.259499e-210 1.714377e-211
```

```r
apply(training%>%filter(HeartDisease==0)%>%select(where(is.numeric)),2,function(x) shapiro.test(x)$p.val
```

```
##          Age    RestingBP        MaxHR      Oldpeak  Cholesterol
## 2.189069e-01 1.764890e-06 1.792072e-03 5.761538e-21 2.095663e-09
```

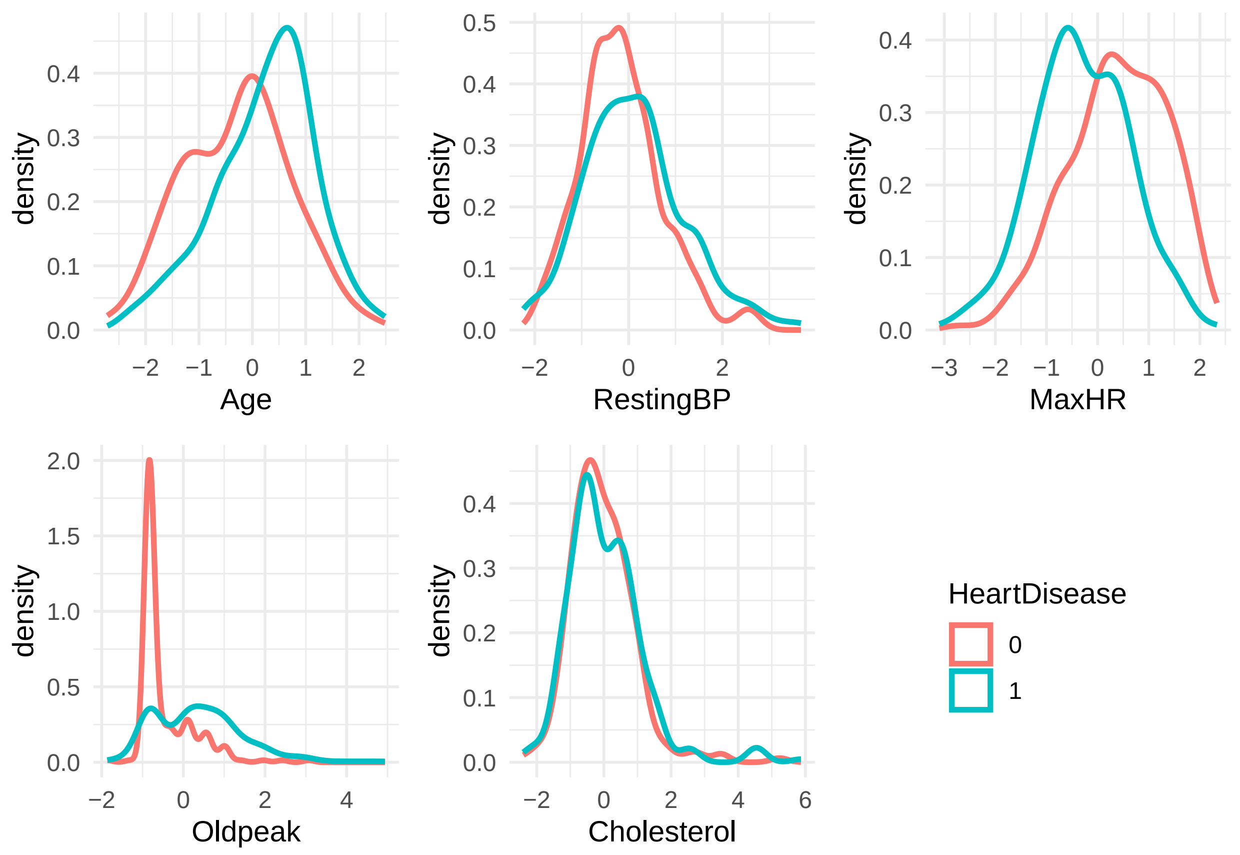
The tests exclude the normality of the conditional distributions of the covariates. Consequently, the application of both linear and quadratic discriminant analysis models will not be possible.

We inspect the densities of the independent variables conditional on the target.

```r
age_dens=ggplot(training,aes(x=Age,col=HeartDisease))+
  geom_density(linewidth=1,show.legend = F)+
  theme_minimal()
resting_dens=ggplot(training,aes(x=RestingBP,col=HeartDisease))+
  geom_density(linewidth=1,show.legend = F)+
  theme_minimal()
maxhr_dens=ggplot(training,aes(x=MaxHR,col=HeartDisease))+
  geom_density(linewidth=1,show.legend = F)+
  theme_minimal()
oldpeak_dens=ggplot(training,aes(x=Oldpeak,col=HeartDisease))+
  geom_density(linewidth=1,show.legend = F)+
  theme_minimal()
choles_dens=ggplot(training,aes(x=Cholesterol,col=HeartDisease))+
  geom_density(linewidth=1,show.legend = F)+
  theme_minimal()
legend=get_legend(ggplot(training,aes(x=Age,col=HeartDisease))+
  geom_density(linewidth=1)+
  theme_minimal())
```

```
## Warning in get_plot_component(plot, "guide-box"): Multiple components found;
## returning the first one. To return all, use `return_all = TRUE`.
```

```r
plot_grid(age_dens,resting_dens,maxhr_dens,oldpeak_dens,choles_dens,legend,ncol=3)
```



The conditionals are clearly multimodal so the assumption of normality is untenable.

Modeling

We are in the heart of the analysis. We consider several models, for each of which we will have a classifier training phase and a validation phase to select the best model among them. The most appropriate evaluation metric for the problem at hand is *Sensitivity*, which indicates the percentage of correctly classified patients.

This is preferred to *Accuracy* because in a medical context the correct detection of disease takes higher priority. In order to maintain an overview of the forecasting quality, we examine both. The two models with the highest prediction performance will be taken to the testing phase.

MDA

Linear Discriminant Analysis and Quadratic Discriminant Analysis assume a Gaussian mixture, i.e. that the covariates distribute normally conditional on the mode of the target.

Mixture Discriminant Analysis (*MDA*) is a classification method that extends the approach of *LDA* and *QDA*, assuming that the conditional distributions of the covariates are themselves mixtures of Normals.

In our case, it can be seen that almost no variables respect the assumptions of *LDA* and *QDA*. We can assume that this does not apply to the *MDA* given the multimodality of the conditional distributions. The *MDA* attempts to estimate the parameters of internal mixtures and the one containing them; for classification purposes it also makes use of a posteriori probabilities.

The complexity of the model assumed implies a large number of parameters to be estimated. For this reason, strong assumptions are usually made on the components of the mixtures within each covariate, such as homoschedasticity.

```r
mod_mda=MclustDA(training[,-12]%>%select(where(is.numeric)),class = training$HeartDisease,G=c(1:9))
pred_mda=predict(mod_mda,validation[,-12]%>%select(where(is.numeric)))
confmat_mda=table(pred_mda$classification,validation$HeartDisease)
metriche(confmat_mda)
```

```
##    Accuracy Sensitivity
##   0.7213115   0.7010309
```

In order to quantify the performance of this model, we need a comparison with the following ones.

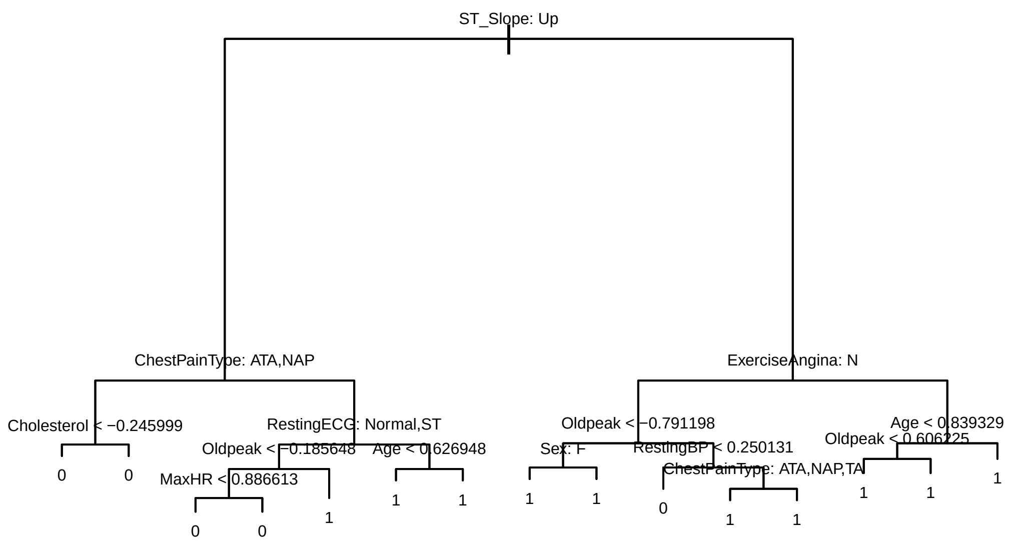
Classification Trees

Classification trees do not require the imputation of missing values as they can deal directly with missing values.

```r
set.seed(8)
tree.Heart = tree(HeartDisease ~.,data=training_original)
summary(tree.Heart)
```

```
##
## Classification tree:
## tree(formula = HeartDisease ~ ., data = training_original)
## Variables actually used in tree construction:
##  [1] "ST_Slope"       "ChestPainType"  "Cholesterol"    "RestingECG"
##  [5] "Oldpeak"        "MaxHR"          "Age"            "ExerciseAngina"
##  [9] "Sex"            "RestingBP"
## Number of terminal nodes:  15
## Residual mean deviance:  0.5842 = 252.4 / 432
## Misclassification error rate: 0.1298 = 58 / 447
```

```r
plot(tree.Heart)
text(tree.Heart,cex=0.5,pretty = 0)
```



The variable *ST_Slope* creates the first branch, from which we deduce that it is very relevant for classification purposes.

```
tree.pred = predict(tree.Heart , validation_original , type = "class")
confmat_tree=table(tree.pred , validation_original$HeartDisease)
metriche(confmat_tree)
```

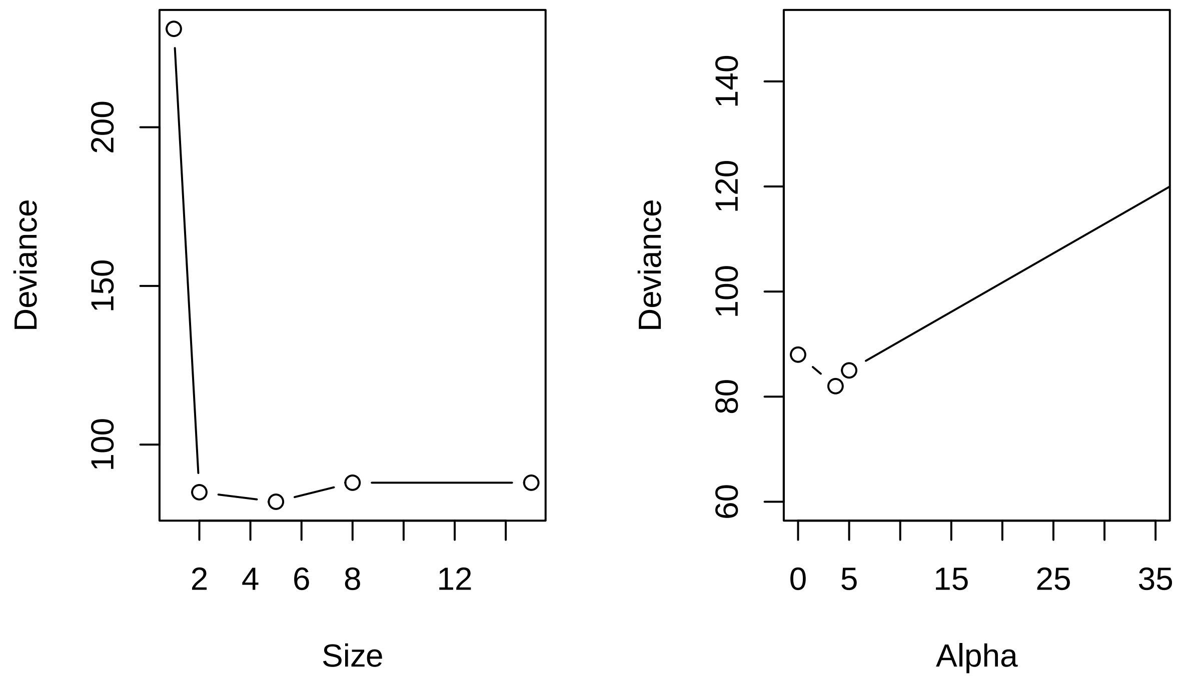
```
##    Accuracy Sensitivity
##   0.7978142   0.7938144
```

We can see an improvement for both evaluation metrics.

Pruning

Classification trees notoriously tend to be too complex, so a technique called *pruning* is applied to simplify the pre-existing tree. Here we opt for *cost complexity pruning*.

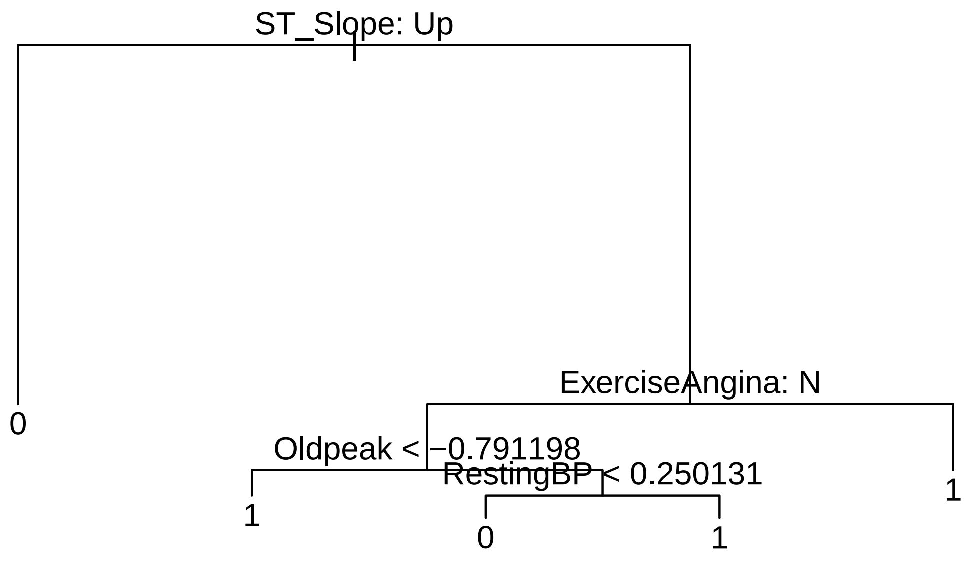
```
cv.Heart = cv.tree(tree.Heart , FUN = prune.misclass)
par(mfrow = c(1, 2))
plot(cv.Heart$size , cv.Heart$dev,ylab="Deviance",xlab="Size", type = "b")
plot(cv.Heart$k, cv.Heart$dev,ylab="Deviance",xlab="Alpha",ylim=c(60,150),xlim=c(0,35) ,type = "b")
```



```
par(mfrow = c(1, 1))
```

```
best_size=cv.Heart$size[which.min(cv.Heart$dev)]
prune.Heart = prune.misclass(tree.Heart , best = best_size)

plot(prune.Heart)
text(prune.Heart , pretty = 0)
```



```
tree.pred_pruning = predict(prune.Heart , validation_original , type = "class")
confmat_pruning=table(tree.pred_pruning , validation_original$HeartDisease)
metriche(confmat_pruning)
```

```
##    Accuracy Sensitivity
##   0.7978142   0.7319588
```

Contrary to what we would have expected, the forecast did not improve in terms of *Sensitivity*.

K-NN

A further classification method is the K-NN. For the calculation of distances between observations, the Euclidean distance is used, which is why we code qualitative variables using *dummies*.

```
train_dummy=training%>%mutate(Sex=ifelse(Sex=="M",1,0),ChestPainTypeATA=ifelse(ChestPainType=="ATA",1,0)
                           ChestPainTypeTA=ifelse(ChestPainType=="TA",1,0),RestingECGNormal=ifelse(Resti
                           RestingECGST=ifelse(RestingECG=="ST",1,0),ExerciseAngina=ifelse(ExerciseAngi
                           ST_SlopeFlat=ifelse(ST_Slope=="Flat",1,0),ST_SlopeUp=ifelse(ST_Slope=="Up",1
train_dummy=train_dummy[,-c(3,6,10)]
valid_dummy=validation%>%mutate(Sex=ifelse(Sex=="M",1,0),ChestPainTypeATA=ifelse(ChestPainType=="ATA",1
                                ChestPainTypeTA=ifelse(ChestPainType=="TA",1,0),RestingECGNormal=ifelse
                                RestingECGST=ifelse(RestingECG=="ST",1,0),ExerciseAngina=ifelse(Exercis
                                ST_SlopeFlat=ifelse(ST_Slope=="Flat",1,0),ST_SlopeUp=ifelse(ST_Slope=="
valid_dummy=valid_dummy[,-c(3,6,10)]
```

```
ks=seq(1,21,by=2)
knn_cv=apply(as.matrix(ks),1,function(x){mod_knn=knn(train_dummy[,-8],k=x,test=valid_dummy[,-8],cl=trai
confmat_knn_cv=table(mod_knn,valid_dummy$HeartDisease)
c(x,confmat_knn_cv[2,2]/sum(confmat_knn_cv[,2]),sum(diag(confmat_knn_cv))/nrow(validation))})
```

The choice of hyper-parameter K is made through cross-validation.

```
k_cv=knn_cv[1,which.max(knn_cv[2,])]
mod_knn_cv=knn(train_dummy[,-8],k=k_cv,test=valid_dummy[,-8],cl=training$HeartDisease)
confmat_knn=table(mod_knn_cv,validation$HeartDisease)
```

```
metriche(confmat_knn)
```

```
##    Accuracy Sensitivity
##   0.8743169   0.8762887
```

The two evaluation metrics taken into account show excellent results.

Logistic Regression

In this section, we will use logistic regression as a classification method.

We begin by evaluating a logistic regression by including all available variables.

```
logistic_mod=glm(HeartDisease~. ,data = training, family="binomial")
summary(logistic_mod)
```

```
##
## Call:
## glm(formula = HeartDisease ~ ., family = "binomial", data = training)
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -0.58025    0.64808  -0.895 0.370612
## Age                0.12469    0.15077   0.827 0.408250
## SexM               1.44424    0.34662   4.167 3.09e-05 ***
## ChestPainTypeATA -1.70506    0.40559  -4.204 2.62e-05 ***
## ChestPainTypeNAP -1.89691    0.34417  -5.512 3.56e-08 ***
## ChestPainTypeTA  -1.25037    0.51106  -2.447 0.014420 *
## RestingBP          0.12333    0.14038   0.879 0.379671
## FastingBS1         1.53934    0.33631   4.577 4.71e-06 ***
## RestingECGNormal -0.27629    0.32763  -0.843 0.399059
## RestingECGST     -0.37777    0.42794  -0.883 0.377363
## MaxHR             -0.05102    0.15721  -0.325 0.745543
## ExerciseAnginaY    1.08680    0.30028   3.619 0.000295 ***
## Oldpeak            0.35203    0.15817   2.226 0.026037 *
## ST_SlopeFlat       1.03106    0.55956   1.843 0.065384 .
## ST_SlopeUp        -1.20726    0.59350  -2.034 0.041939 *
## Cholesterol        0.00913    0.12664   0.072 0.942524
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 757.32  on 551  degrees of freedom
## Residual deviance: 382.13  on 536  degrees of freedom
## AIC: 414.13
##
## Number of Fisher Scoring iterations: 5
```

We now evaluate the logistic regression for *step* according to the *AIC* criterion.

```
step_logistic_mod=step(logistic_mod, direction = "both", trace = F)
summary(step_logistic_mod)
```

```
##
## Call:
## glm(formula = HeartDisease ~ Sex + ChestPainType + FastingBS +
##     ExerciseAngina + Oldpeak + ST_Slope, family = "binomial",
##     data = training)
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -0.7655     0.6120  -1.251   0.2110
## SexM               1.3762     0.3401   4.046 5.20e-05 ***
## ChestPainTypeATA  -1.7261     0.3978  -4.339 1.43e-05 ***
## ChestPainTypeNAP  -1.8586     0.3369  -5.517 3.46e-08 ***
## ChestPainTypeTA   -1.1331     0.5016  -2.259   0.0239 *
## FastingBS1         1.5230     0.3308   4.604 4.13e-06 ***
## ExerciseAnginaY    1.1552     0.2859   4.040 5.34e-05 ***
## Oldpeak            0.3746     0.1540   2.432   0.0150 *
## ST_SlopeFlat       1.0293     0.5524   1.864   0.0624 .
## ST_SlopeUp        -1.2552     0.5808  -2.161   0.0307 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 757.32  on 551  degrees of freedom
## Residual deviance: 385.67  on 542  degrees of freedom
## AIC: 405.67
##
## Number of Fisher Scoring iterations: 5
```

```
anova(step_logistic_mod,test = "Chisq")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: HeartDisease
##
## Terms added sequentially (first to last)
```

```
##
##
##                Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                            551     757.32
## Sex             1   47.523       550     709.80 5.437e-12 ***
## ChestPainType   3  141.032       547     568.77 < 2.2e-16 ***
## FastingBS       1   25.878       546     542.89 3.637e-07 ***
## ExerciseAngina  1   62.197       545     480.70 3.108e-15 ***
## Oldpeak         1   30.422       544     450.27 3.475e-08 ***
## ST_Slope        2   64.601       542     385.67 9.378e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We note that in the *step_logistic_mod* model there is an improvement in the *Akaike* criterion by removing the variables **Cholesterol** and **MaxHR**.

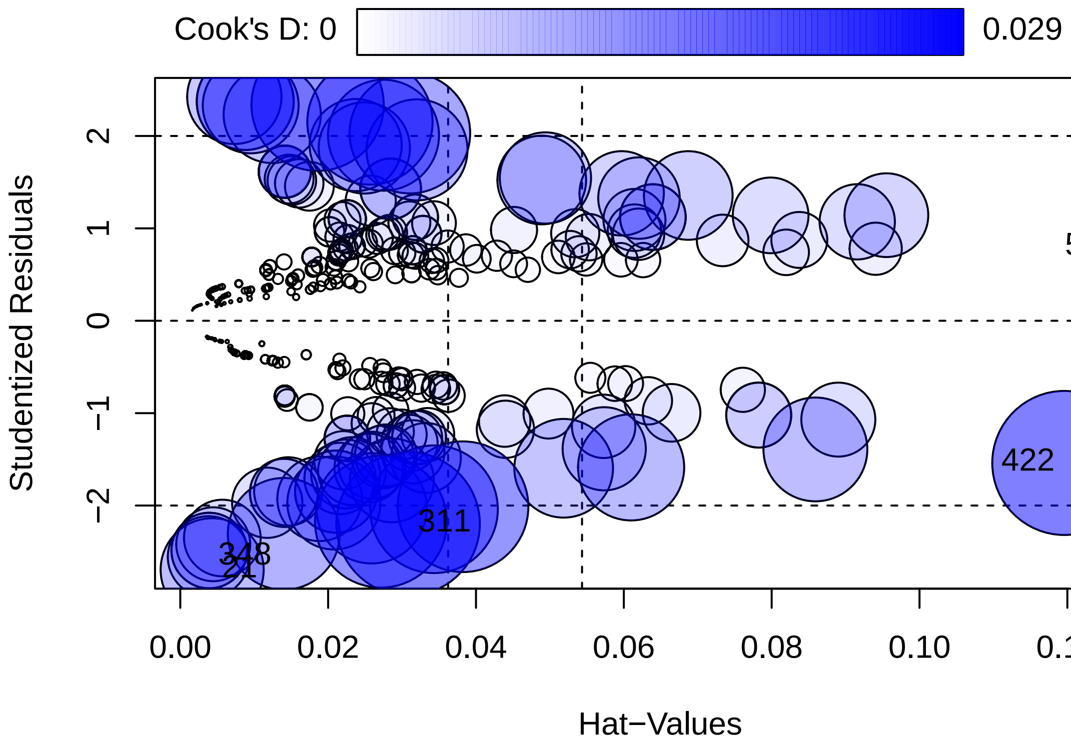
We check for outliers and influence points

```
outlierTest(step_logistic_mod)
```

```
## No Studentized residuals with Bonferroni p < 0.05
## Largest |rstudent|:
##     rstudent unadjusted p-value Bonferroni p
## 21 -2.694785          0.0070434           NA
```

No outliers detected

```
infplot=influencePlot(step_logistic_mod)
```



Let us try to remove the influential points highlighted by the graph, bearing in mind an initial regression model containing all variables.

```
obs_inf=as.numeric(rownames(infplot))
logistic_mod_no_inf=glm(HeartDisease~. ,data = training[-obs_inf,], family="binomial")
summary(logistic_mod_no_inf)
```

```
##
## Call:
## glm(formula = HeartDisease ~ ., family = "binomial", data = training[-obs_inf,
##     ])
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -0.30181    0.72068  -0.419 0.675375
## Age                0.12077    0.15484   0.780 0.435423
## SexM               1.59413    0.36088   4.417 9.99e-06 ***
## ChestPainTypeATA -1.75052    0.41453  -4.223 2.41e-05 ***
## ChestPainTypeNAP -2.03019    0.35793  -5.672 1.41e-08 ***
## ChestPainTypeTA  -1.18634    0.52840  -2.245 0.024759 *
## RestingBP          0.15403    0.14705   1.047 0.294908
## FastingBS1         1.58212    0.35030   4.516 6.29e-06 ***
## RestingECGNormal -0.24244    0.33708  -0.719 0.471997
## RestingECGST     -0.41805    0.44706  -0.935 0.349729
## MaxHR             -0.10468    0.16349  -0.640 0.522008
## ExerciseAnginaY    1.07210    0.31107   3.446 0.000568 ***
## Oldpeak           0.44338    0.17387   2.550 0.010769 *
## ST_SlopeFlat      0.71780    0.63037   1.139 0.254833
## ST_SlopeUp       -1.54978    0.65402  -2.370 0.017806 *
## Cholesterol       0.09341    0.13264   0.704 0.481293
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 749.58  on 546  degrees of freedom
## Residual deviance: 359.18  on 531  degrees of freedom
## AIC: 391.18
##
## Number of Fisher Scoring iterations: 6
```

We notice an improvement in terms of *AIC* compared to the two previous models.

As before, we apply a *stepwise* selection to the model without influence points.

```
step_logistic_mod_no_inf=step(logistic_mod_no_inf, direction = "both", trace = F)
summary(step_logistic_mod_no_inf)
```

```
##
## Call:
## glm(formula = HeartDisease ~ Sex + ChestPainType + FastingBS +
##     ExerciseAngina + Oldpeak + ST_Slope, family = "binomial",
##     data = training[-obs_inf, ])
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -0.5321     0.6804  -0.782  0.43420
## SexM                1.5021     0.3534   4.250 2.14e-05 ***
## ChestPainTypeATA   -1.7695     0.4058  -4.361 1.30e-05 ***
## ChestPainTypeNAP   -1.9839     0.3491  -5.683 1.32e-08 ***
## ChestPainTypeTA    -1.0858     0.5235  -2.074  0.03806 *
## FastingBS1          1.5521     0.3435   4.518 6.24e-06 ***
## ExerciseAnginaY     1.1813     0.2954   3.999 6.37e-05 ***
## Oldpeak             0.4379     0.1696   2.581  0.00985 **
## ST_SlopeFlat        0.7945     0.6216   1.278  0.20118
## ST_SlopeUp         -1.5678     0.6424  -2.441  0.01467 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 749.58  on 546  degrees of freedom
## Residual deviance: 363.82  on 537  degrees of freedom
## AIC: 383.82
##
## Number of Fisher Scoring iterations: 6
```

```
anova(step_logistic_mod_no_inf,test = "Chisq")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: HeartDisease
##
## Terms added sequentially (first to last)
##
##
##                 Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                              546     749.58
## Sex              1   50.473       545     699.10 1.208e-12 ***
## ChestPainType    3  143.734       542     555.37 < 2.2e-16 ***
## FastingBS        1   26.597       541     528.77 2.506e-07 ***
## ExerciseAngina   1   64.373       540     464.40 1.030e-15 ***
## Oldpeak          1   35.775       539     428.62 2.214e-09 ***
## ST_Slope         2   64.802       537     363.82 8.481e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We observe a further reduction in *AIC*.

We now compare the behaviour of the 4 logistic regression models found according to the metrics of evaluation chosen.

```r
step_logistic_mod_no_inf_pred = round(predict(step_logistic_mod_no_inf, validation, type = "response"),
confmat_step_logistic_mod_no_inf=table(step_logistic_mod_no_inf_pred , validation$HeartDisease)
```

Stepwise regression without influencers

```r
metriche(confmat_step_logistic_mod_no_inf)
```

```
##    Accuracy Sensitivity
##   0.8797814   0.8762887
```

```r
step_logistic_mod_pred = round(predict(step_logistic_mod, validation, type = "response"),0)
confmat_step_logistic_mod=table(step_logistic_mod_pred , validation$HeartDisease)
```

Stepwise regression

```r
metriche(confmat_step_logistic_mod)
```

```
##    Accuracy Sensitivity
##   0.8743169   0.8659794
```

```r
logistic_mod_pred = round(predict(logistic_mod, validation, type = "response"),0)
confmat_logistic_mod=table(logistic_mod_pred , validation$HeartDisease)
```

Regression

```
metriche(confmat_logistic_mod)
```

```
##    Accuracy Sensitivity
##   0.8743169   0.8556701
```

```
logistic_mod_no_inf_pred = round(predict(logistic_mod_no_inf, validation, type = "response"),0)
confmat_logistic_mod_no_inf=table(logistic_mod_no_inf_pred , validation$HeartDisease)
```

Regression without influencers

```
metriche(confmat_logistic_mod_no_inf)
```

```
##    Accuracy Sensitivity
##   0.8797814   0.8659794
```

Logistic regression is highly effective in this context, the removal of influential points has little impact on the metrics. We prefer the *stepwise* model because it contains fewer features and is therefore less complex.

Regression Model Diagnostics For diagnostics, we consider the regression model with the best performance levels, i.e. the one with stepwise criteria without influence points.

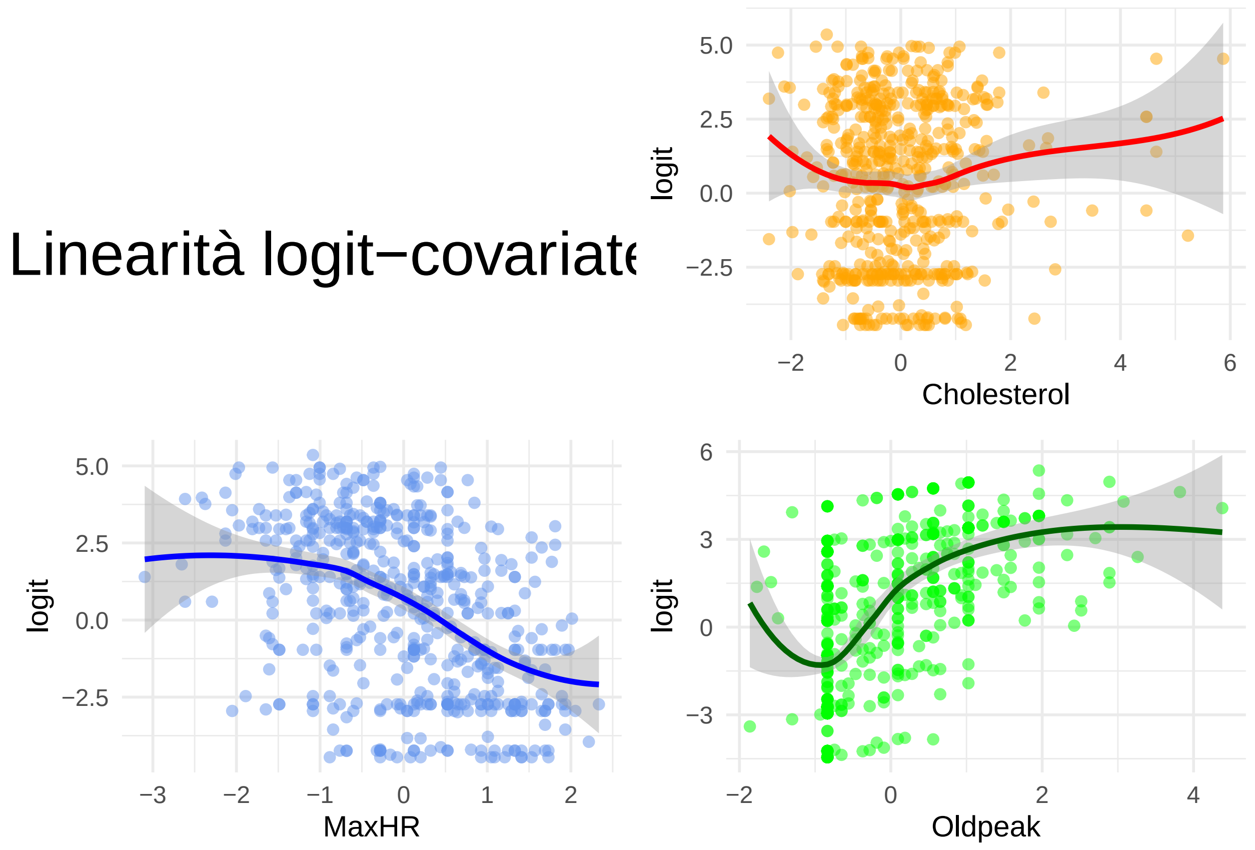
We plot the trend of the target variable, appropriately transformed (logit), with that of the numerical covariates. (Qualitative variables are meaningless to represent).

```
probabilities = predict(step_logistic_mod_no_inf, type = "response")
predictors = c("Sex","ChestPainType","Cholesterol","FastingBS","MaxHR","ExerciseAngina","Oldpeak","ST_Sl
logit_training=training[-obs_inf,]%>%
  select(all_of(predictors))
logit_training = logit_training %>%
  mutate(logit = log(probabilities/(1-probabilities)))

text = paste("\n Linearità logit-covariate")

plot_grid(ggplot() +
  annotate("text", x = 4, y = 25, size=8, label = text) +
  theme_void(),
  ggplot(data=logit_training, aes(Cholesterol,logit))+
  geom_point(alpha=.5,col="orange")+
  geom_smooth(method="loess",col="red")+
  theme_minimal(),
ggplot(data=logit_training, aes(MaxHR,logit))+
  geom_point(alpha=.5,col="cornflowerblue")+
  geom_smooth(method="loess",col="blue")+
  theme_minimal(),
ggplot(data=logit_training, aes(Oldpeak,logit))+
  geom_point(alpha=.5,col="green")+
  geom_smooth(method="loess",col="darkgreen")+
  theme_minimal()
)
```

```
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
```



The assumption of linearity is sufficiently verified.

Model selection

Let us now compare the performance of the models considered, to do so we use additional metrics to those used so far for better understanding.

```
confmats=list(mda=confmat_mda,tree=confmat_tree,pruning=confmat_pruning
              ,knn=confmat_knn,reglog=confmat_logistic_mod,reglognf=confmat_logistic_mod_no_inf,
              regstep=confmat_step_logistic_mod,regstepnf=confmat_step_logistic_mod_no_inf)
sapply(confmats,metriche)
```

```
##                   mda      tree   pruning       knn    reglog  reglognf
## Accuracy    0.7213115 0.7978142 0.7978142 0.8743169 0.8743169 0.8797814
## Sensitivity 0.7010309 0.7938144 0.7319588 0.8762887 0.8556701 0.8659794
##               regstep regstepnf
## Accuracy    0.8743169 0.8797814
## Sensitivity 0.8659794 0.8762887
```

F1-Score

```
f1_score=function(tab){
  Recall=tab[2,2]/sum(tab[,2])
  Precision=tab[2,2]/sum(tab[2,])
  f1=2*Recall*Precision/(Recall+Precision)
  return(f1)
}
sapply(confmats,f1_score)
```

```
##       mda      tree   pruning       knn    reglog  reglognf   regstep regstepnf
## 0.7272727 0.8062827 0.7932961 0.8808290 0.8783069 0.8842105 0.8795812 0.8854167
```

From the calculated metrics we deduce that we can exclude the tree, pruned and unpruned, and the *MDA* from the analysis on the test set. For the logistic regressions the *F1-Score* also shows little difference.

ROC curve

```
prediction_reglog=prediction(step_logistic_mod_pred,validation$HeartDisease)
performance(prediction_reglog,measure="auc")@y.values
```

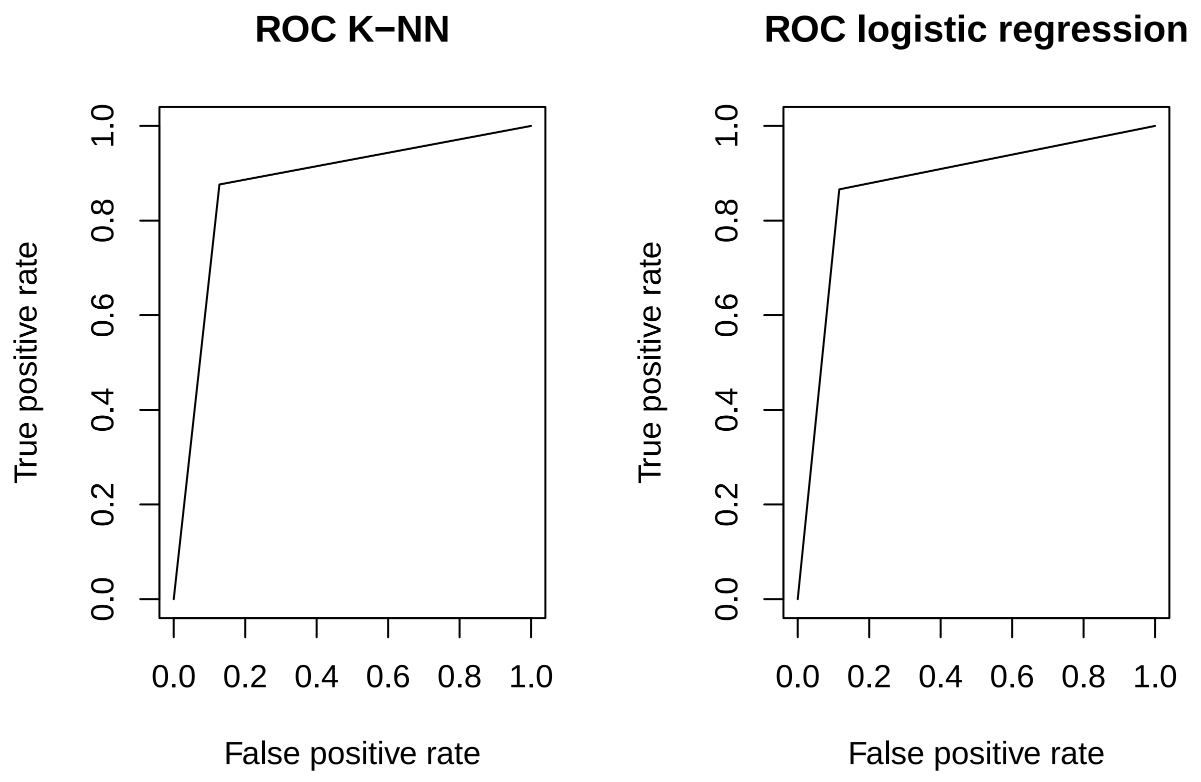
```
## [[1]]
## [1] 0.8748502
```

```
knn.pred=as.numeric(as.vector(mod_knn_cv))
prediction_knn=prediction(knn.pred,validation$HeartDisease)
performance(prediction_knn,measure="auc")@y.values
```

```
## [[1]]
## [1] 0.8741908
```

```
par(mfrow=c(1,2))
plot(performance(prediction_knn,"tpr","fpr"),main="ROC K-NN")

plot(performance(prediction_reglog,"tpr","fpr"),main="ROC logistic regression")
```



```
par(mfrow=c(1,1))
```

These two indicators do not show substantial differences between the two models, so we will take both to the testing phase.

Testing

In the testing phase we merge the Training and Validation Set into a single data set, on which the models will be re-trained.

We again standardise and treat the missing values as before.

```
big_training=rbind(training_non_stand,validation_non_stand)
medie=colMeans(big_training%>%select(where(is.numeric)),na.rm=T)
varianze=apply(big_training%>%select(where(is.numeric)),2,function(x){var(x,na.rm=T)})
big_training[,c("Age","RestingBP","Cholesterol","MaxHR","Oldpeak")]=scale(big_training[,c("Age","Restin
test_non_stand[,c("Age","RestingBP","Cholesterol","MaxHR","Oldpeak")]=scale(test_non_stand[,c("Age","Re
final.test=test_non_stand
heart_non_stand[,c("Age","RestingBP","Cholesterol","MaxHR","Oldpeak")]=scale(heart_non_stand[,c("Age","
final.heart=heart_non_stand
```

```
heart_imputato=mice(final.heart,predictorMatrix = matrix,meth="cart",m=1,printFlag = F,maxit = 50,ignor
final.test$Cholesterol=complete(heart_imputato)[test.index,"Cholesterol"]
big_training=inner_join(big_training,complete(heart_imputato),join_by(Age,Sex,ChestPainType,RestingBP,F
  mutate(Cholesterol=Cholesterol.y)%>%
  select(!contains("."))
```

Logistic regression and K-NN models are being tested.

Logistic regression :

```
reg_log=glm(HeartDisease~. ,data = big_training, family="binomial")
summary(reg_log)
```

```
##
## Call:
## glm(formula = HeartDisease ~ ., family = "binomial", data = big_training)
##
## Coefficients:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -0.57264    0.57706  -0.992 0.321028
## Age               0.21309    0.13740   1.551 0.120927
## SexM              1.63526    0.31740   5.152 2.58e-07 ***
## ChestPainTypeATA -1.97904    0.36222  -5.464 4.66e-08 ***
## ChestPainTypeNAP -1.92523    0.30005  -6.416 1.39e-10 ***
## ChestPainTypeTA  -1.55313    0.46372  -3.349 0.000810 ***
## RestingBP         0.01185    0.12329   0.096 0.923463
## FastingBS1        1.41912    0.30103   4.714 2.43e-06 ***
## RestingECGNormal -0.16448    0.29570  -0.556 0.578053
## RestingECGST     -0.35277    0.38451  -0.917 0.358905
## MaxHR            -0.13900    0.13925  -0.998 0.318196
## ExerciseAnginaY   0.98547    0.27301   3.610 0.000307 ***
## Oldpeak           0.30846    0.13789   2.237 0.025286 *
## ST_SlopeFlat      0.98951    0.49241   2.010 0.044479 *
## ST_SlopeUp       -1.32184    0.51846  -2.550 0.010787 *
## Cholesterol       0.08853    0.12051   0.735 0.462585
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1010.84  on 734  degrees of freedom
## Residual deviance:  485.26  on 719  degrees of freedom
## AIC: 517.26
##
## Number of Fisher Scoring iterations: 6
```

```
step_reg_log=step(reg_log, direction = "both", trace = F)
summary(step_reg_log)
```

```
##
## Call:
## glm(formula = HeartDisease ~ Age + Sex + ChestPainType + FastingBS +
##     ExerciseAngina + Oldpeak + ST_Slope, family = "binomial",
##     data = big_training)
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -0.7021     0.5469  -1.284 0.199229
## Age                 0.2578     0.1225   2.105 0.035277 *
## SexM                1.5963     0.3120   5.115 3.13e-07 ***
## ChestPainTypeATA   -2.0416     0.3579  -5.705 1.16e-08 ***
## ChestPainTypeNAP   -1.9532     0.2960  -6.599 4.14e-11 ***
## ChestPainTypeTA    -1.5852     0.4581  -3.461 0.000539 ***
## FastingBS1          1.3870     0.2974   4.664 3.10e-06 ***
## ExerciseAnginaY     1.0171     0.2623   3.878 0.000105 ***
## Oldpeak             0.2954     0.1348   2.191 0.028450 *
## ST_SlopeFlat        1.0242     0.4864   2.106 0.035228 *
## ST_SlopeUp         -1.3444     0.5105  -2.634 0.008450 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1010.84  on 734  degrees of freedom
## Residual deviance:  487.35  on 724  degrees of freedom
## AIC: 509.35
##
## Number of Fisher Scoring iterations: 5
```

```
outlierTest(step_reg_log)
```

```
## No Studentized residuals with Bonferroni p < 0.05
## Largest |rstudent|:
##     rstudent unadjusted p-value Bonferroni p
## 21 -2.680904          0.0073424           NA
```

Again, we do not detect any outliers.

```
confmat_test_stepreglog=table(round(predict(step_reg_log,final.test,type="response")),final.test$HeartD
```

Logistic regression with *stepwise* selection

```
metriche(confmat_test_stepreglog)
```

```
##    Accuracy Sensitivity
##   0.8846154   0.9108911
```

```
obs_influenti=as.numeric(rownames(influence_plot))
step_reg_log_no_inf<-glm(HeartDisease~. ,data = training[-obs_influenti,], family="binomial")
```

Logistic regression with *stepwise* selection without influential points.

```
confmat_test_stepreglog_noinf=table(round(predict(step_reg_log_no_inf,final.test,type="response")),final
metriche(confmat_test_stepreglog_noinf)
```

```
##    Accuracy Sensitivity
##   0.8791209   0.9009901
```

K-Nearest Neighbours:

```
big_training_dummy=big_training%>%mutate(Sex=ifelse(Sex=="M",1,0),ChestPainTypeATA=ifelse(ChestPainType=
       ChestPainTypeTA=ifelse(ChestPainType=="TA",1,0),RestingECGNormal=ifelse(RestingECG=="Normal",1,0)
       RestingECGST=ifelse(RestingECG=="ST",1,0),ExerciseAngina=ifelse(ExerciseAngina=="Y",1,0),
       ST_SlopeFlat=ifelse(ST_Slope=="Flat",1,0),ST_SlopeUp=ifelse(ST_Slope=="Up",1,0))
big_training_dummy=big_training_dummy[,-c(3,6,10)]

final_test_dummy=final.test%>%mutate(Sex=ifelse(Sex=="M",1,0),ChestPainTypeATA=ifelse(ChestPainType=="A
                                    ChestPainTypeTA=ifelse(ChestPainType=="TA",1,0),RestingECGNormal=ifel
                                    RestingECGST=ifelse(RestingECG=="ST",1,0),ExerciseAngina=ifelse(Exerc
                                    ST_SlopeFlat=ifelse(ST_Slope=="Flat",1,0),ST_SlopeUp=ifelse(ST_Slope=
final_test_dummy=final_test_dummy[,-c(3,7,11)]

knn_cv=knn(big_training_dummy[,-8],k=k_cv,test=final_test_dummy[,-9],cl=big_training_dummy$HeartDisease)
confmat_knn_cv=table(knn_cv,final_test_dummy$HeartDisease)
metriche(confmat_knn_cv)
```

```
##    Accuracy Sensitivity
##   0.7802198   0.8514851
```

Comparing the *Accuracy* and *Sensitivity* metrics of the two classifiers, we deduce that the model that best classifies the test set is logistic regression selected according to the *Stepwise* criterion.

Model analysis

In this last section we subject the models we found best during the modelling phase to analysis. In particular, we observe whether the models may have overfitted. We then check the accuracy in both the Test and Training Set.

```
knn.train.pred=knn(big_training_dummy[,-8],k=k_cv,test=big_training_dummy[,-8],cl=big_training_dummy$He
confmat_knn_train=table(knn.train.pred , big_training_dummy$HeartDisease)
```

Accuracy K-NN in training

```
metriche(confmat_knn_train)[1]
```

```
## Accuracy
## 0.862585
```

Accuracy K-NN in test

```r
metriche(confmat_knn_cv)[1]
```

```
##  Accuracy
## 0.7802198
```

Accuracy Logistic regression *stepwise* in training

```r
confmat_train_stepreglog=table(round(predict(step_reg_log,big_training,type="response")),big_training$He
metriche(confmat_train_stepreglog)[1]
```

```
##  Accuracy
## 0.8598639
```

Accuracy Logistic regression *stepwise* in the test

```r
metriche(confmat_test_stepreglog)[1]
```

```
##  Accuracy
## 0.8846154
```

We note that K-NN has a significantly higher Error rate in the Test set than Training. This observation suggests that it is overfitting. In contrast, the logistic regression model from the Training to the Test set improves by a few percentage points, ruling out a possible overfitting scenario.

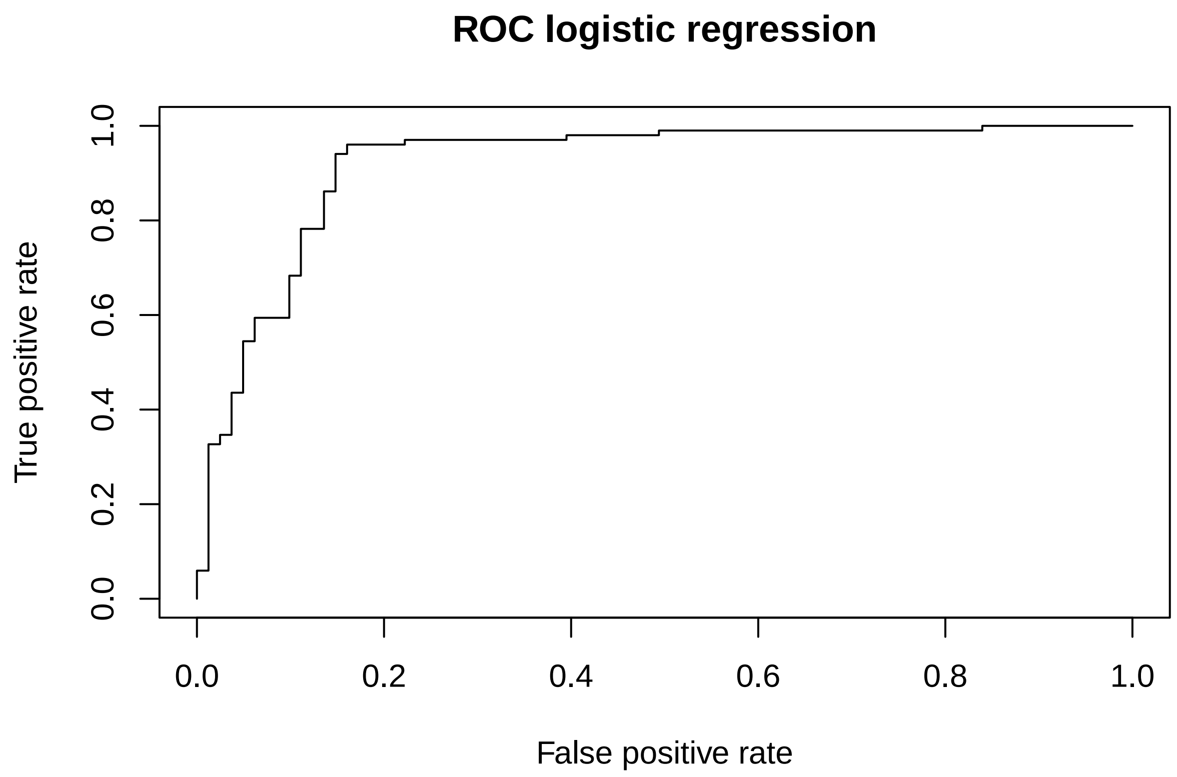
Test evaluation metrics

Let us now calculate other evaluation metrics to better delineate the comparison between the two models.

```r
prediction_test_reglog=prediction(predict(step_reg_log,final.test,type="response"),final.test$HeartDisea
performance(prediction_test_reglog,measure="auc")@y.values
```

```
## [[1]]
## [1] 0.919692
```

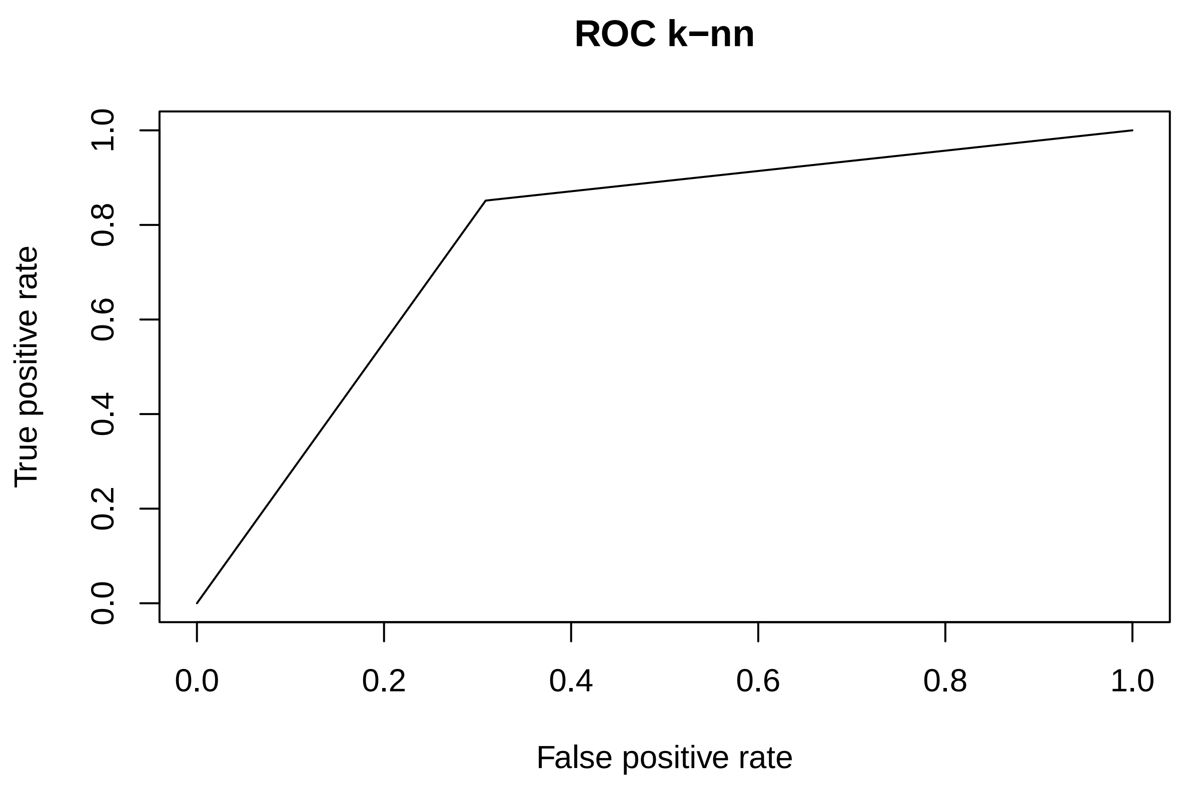
```r
plot(performance(prediction_test_reglog,"tpr","fpr"),main="ROC logistic regression")
```



```
prediction_knn=prediction(as.numeric(knn_cv)-1,final.test$HeartDisease)
performance(prediction_knn,measure="auc")@y.values
```

```
## [[1]]
## [1] 0.7714216
```

```
plot(performance(prediction_knn,"tpr","fpr"),main="ROC k-nn")
```



Looking at the AUC values, we can see that the logistic regression model prevails in the comparison with K-NN.

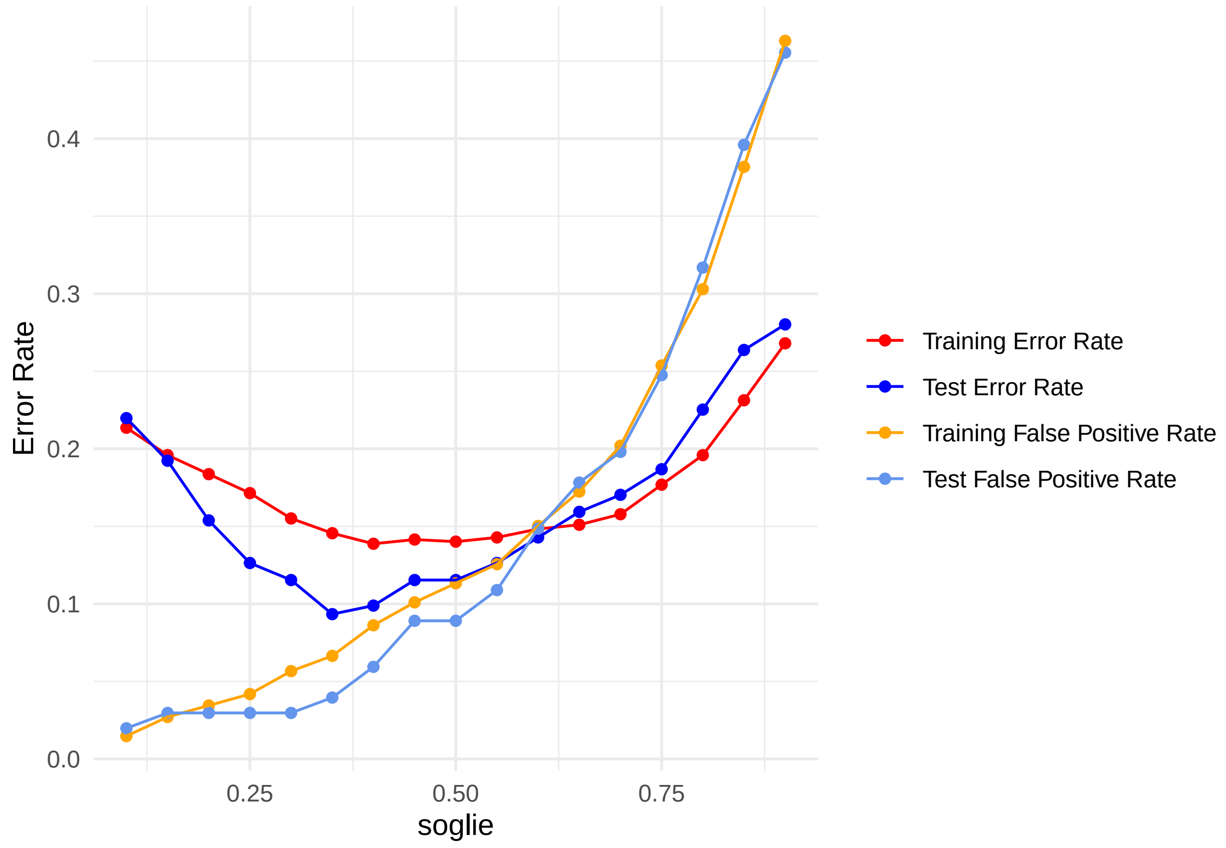
Threshold

So far we have used a threshold for the a posteriori probability of 0.5, there is no guarantee that this is the best. We dedicate ourselves to finding the optimal threshold by looking at the error and false positive rates on the training and test set.

```
test_train_error=function(soglia,modello=step_reg_log){
  trpred=predict(modello,big_training,type="response")
  testpred=predict(modello,final.test,type="response")
  ptr=ifelse(trpred>soglia,yes = 1,no=0)
  ptest=ifelse(testpred>soglia,1,0)
  met1=1-metriche(table(ptr,big_training$HeartDisease))[1]
  met2=1-metriche(table(ptest,final.test$HeartDisease))[1]
  met3=1-metriche(table(ptr,big_training$HeartDisease))[2]
  met4=1-metriche(table(ptest,final.test$HeartDisease))[2]
  met=rbind(met1,met2,met3,met4)
  rownames(met)=c("training error","test error", "training fp rate","test fp rate")
  colnames(met)="Metriche"
  return(met)
}

test_train_error=Vectorize(test_train_error,vectorize.args = "soglia")
```

```
soglie=seq(0.1,0.9,by=0.05)
errors_soglie=as.data.frame(cbind(soglie,t(test_train_error(soglie))))
colnames(errors_soglie)=c("soglie","training_error","test_error","training_fp_rate","test_fp_rate")
ggplot(errors_soglie,aes(x=soglie))+
  geom_point(aes(y=training_error,col="Training Error Rate"))+
  geom_line(aes(y=training_error,col="Training Error Rate"))+
  geom_point(aes(y=test_error,col="Test Error Rate"))+
  geom_point(aes(y=training_fp_rate,col="Training False Positive Rate"))+
  geom_point(aes(y=test_fp_rate,col="Test False Positive Rate"))+
  geom_line(aes(y=test_error,col="Test Error Rate"))+
  geom_line(aes(y=training_fp_rate,col="Training False Positive Rate"))+
  geom_line(aes(y=test_fp_rate,col="Test False Positive Rate"))+
  ylab("Error Rate")+
  theme_minimal()+
  scale_color_manual(name="",breaks=c("Training Error Rate","Test Error Rate","Training False Positive R
                                                                  "Training False Positive Rate"="orang
```



From the graph, we can deduce that the boundary that guarantees minimum rates on the test is 0.35.

Let us determine further evaluation metrics with this new boundary and test whether it constitutes an improvement in forecast performance.

```
metriche_soglia=cbind(errors_soglie[c(6,9),1],1-errors_soglie[c(6,9),-1])
colnames(metriche_soglia)=c("soglia","training_accuracy","test_accuracy","training_sensitivity","test_se
metriche_soglia
```

```
##   soglia training_accuracy test_accuracy training_sensitivity test_sensitivity
## 6   0.35         0.8544218     0.9065934            0.9334975        0.9603960
## 9   0.50         0.8598639     0.8846154            0.8866995        0.9108911
```

As could be deduced from the graph, we have a substantial increase in the metrics considered.

Let us check whether the improvement is also confirmed by more complex metrics.

```
prob_reg_log=predict(step_reg_log,final.test,type="response")
pred_reg_log_soglia=ifelse(prob_reg_log>0.35,yes = 1,no=0)
confmat_best_soglia=table(pred_reg_log_soglia,final.test$HeartDisease)
f1scores=cbind(f1_score(confmat_best_soglia),f1_score(confmat_test_stepreglog))
colnames(f1scores)=c("F1 score 0.35","F1 score 0.5")
f1scores
```

```
##      F1 score 0.35 F1 score 0.5
## [1,]     0.9194313     0.897561
```

```
prob_reglog=predict(step_logistic_mod,validation,type = "response")
logloss_reglog=-mean(step_logistic_mod_pred*log(prob_reglog)+(1-step_logistic_mod_pred)*log(1-prob_regl

logloss_reglog_soglia=-mean(pred_reg_log_soglia*log(prob_reg_log)+(1-pred_reg_log_soglia)*log(1-prob_reg
c("Log-loss 0.35"=logloss_reglog_soglia,"Log-loss 0.5"=logloss_reglog)
```

```
## Log-loss 0.35  Log-loss 0.5
##     0.1642174     0.1684336
```

Both confirm the improvement given by the shift in the threshold for posterior probabilities.

Conclusions

From the results of the previous section, we can conclude that the best model is the logistic regression model with *stepwise* selection.

```
(summ=summary(step_reg_log))
```

```
##
## Call:
## glm(formula = HeartDisease ~ Age + Sex + ChestPainType + FastingBS +
##     ExerciseAngina + Oldpeak + ST_Slope, family = "binomial",
##     data = big_training)
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -0.7021     0.5469  -1.284 0.199229
## Age                 0.2578     0.1225   2.105 0.035277 *
## SexM                1.5963     0.3120   5.115 3.13e-07 ***
## ChestPainTypeATA   -2.0416     0.3579  -5.705 1.16e-08 ***
## ChestPainTypeNAP   -1.9532     0.2960  -6.599 4.14e-11 ***
## ChestPainTypeTA    -1.5852     0.4581  -3.461 0.000539 ***
## FastingBS1          1.3870     0.2974   4.664 3.10e-06 ***
## ExerciseAnginaY    1.0171     0.2623   3.878 0.000105 ***
## Oldpeak            0.2954     0.1348   2.191 0.028450 *
## ST_SlopeFlat       1.0242     0.4864   2.106 0.035228 *
## ST_SlopeUp        -1.3444     0.5105  -2.634 0.008450 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1010.84  on 734  degrees of freedom
## Residual deviance:  487.35  on 724  degrees of freedom
## AIC: 509.35
##
## Number of Fisher Scoring iterations: 5
```

We now calculate the percentage changes in the odds induced by the variables.

```
(odds_perc_var=(exp(summ$coefficients[-1,1])-1)*100)
```

```
##              Age             SexM ChestPainTypeATA ChestPainTypeNAP
##         29.41259        393.45487        -87.01827        -85.81858
##   ChestPainTypeTA       FastingBS1   ExerciseAnginaY          Oldpeak
##        -79.50947        300.27286        176.51394         34.36955
##     ST_SlopeFlat       ST_SlopeUp
##        178.49593        -73.93046
```

Looking at the percentage variations, it is interesting to note that the male sex seems to be more prone to heart problems. Before coming to conclusions, let us look at the proportions in terms of gender and condition.

```
table(heart$Sex,heart$HeartDisease)
```

```
##
##       0   1
##   F 143  50
##   M 267 457
```

It is evident from the table that there is an imbalance in the gender of the people considered in the study. However, it is medically known that male individuals are more vulnerable to heart disease because oestrogen, which is more prevalent in females, protects against certain heart diseases.

```
table(heart$FastingBS,heart$HeartDisease)
```

```
##
##       0   1
##   0 366 337
##   1  44 170
```

The same argument can be made for the variable **FastingBS** as it is deeply unbalanced in the distribution.

One feature that is certainly important is **ST_Slope**, as its slope is relevant to possible heart disease. From our studies the most favourable condition is a positive ST-slope, in contrast an electrocardiogram showing a flat ST-segment is the most adverse case.

As one might expect, ageing results in an increased odds of experiencing cardiac problems.

```r
(sqrt(varianze[-c(2,3,4)]))
```

```
##      Age  Oldpeak
## 9.569229 1.068906
```

Since the numerical variables were standardized, the increase in the odds ratio is not the result of a one-unit increase in the variable, but rather of an increase equal to its standard deviation. Consequently, the 29.4% increase occurs every 9.6 years.

Another warning sign is exercise-induced chest pain, which represents a significant symptom in the identification of potential heart disease.

Finally, it is interesting to observe that the categories of chest pain perceived as less concerning by patients are typical angina, atypical angina, and non-anginal pain. Curiously, individuals without chest pain turn out to be the ones at highest risk.

The model we estimated after careful analysis showed good adaptability to the test set and excellent predictive capabilities. Although the primary focus during model selection was on its ability to identify diseased individuals, it achieved outstanding all-around performance.

Machine Learning models of this kind have gained significant traction in the biomedical field in recent times, proving to be valid and reliable not only from a theoretical standpoint but also in practical applications.