Heart Disease Prediction Machine Learning Project

Nirmal Sai Swaroop Janapaneedi

may 06 2020

Contents

1. Introduction, overview, executive summary

1.1 Introduction

Heart disease describes a range of conditions that affect your heart. Diseases under the heart disease umbrella include blood vessel diseases, such as coronary artery disease; heart rhythm problems (arrhythmias); and heart defects you're born with (congenital heart defects), among others.

The term **"heart disease"** is often used interchangeably with the term "cardiovascular disease." Cardiovas-cular disease generally refers to conditions that involve narrowed or blocked blood vessels that can lead to a heart attack, chest pain (angina) or stroke. Other heart conditions, such as those that affect your heart's muscle, valves or rhythm, also are considered forms of heart disease. Many forms of heart disease can be prevented or treated with healthy lifestyle choices.

Symptoms of heart disease in your blood vessels (atherosclerotic disease):

Chest pain, chest tightness, chest pressure and chest discomfort (angina) Shortness of breath
Pain, numbness, weakness or coldness in your legs or arms if the blood vessels are narrowed Pain in the neck, jaw, throat, upper abdomen or back

Heart disease symptoms caused by abnormal heartbeats (heart arrhythmias):

Fluttering in your chest
Racing heartbeat (tachycardia)
Slow heartbeat (bradycardia)
Chest pain or discomfort
Shortness of breath
Lightheadedness
Dizziness
Fainting (syncope) or near fainting

Heart disease symptoms caused by heart defects:

Pale gray or blue skin color (cyanosis)
Swelling in the legs, abdomen or areas around the eyes
In an infant, shortness of breath during feedings, leading to poor weight gain
Easily getting short of breath during exercise or activity
Easily tiring during exercise or activity
Swelling in the hands, ankles or feet

Heart disease symptoms caused by weak heart muscle (dilated cardiomyopathy):

Breathlessness with exertion or at rest

Swelling of the legs, ankles and feet
Fatigue
Irregular heartbeats that feel rapid, pounding or fluttering
Dizziness, lightheadedness and fainting

Heart disease symptoms caused by heart infections:

Fever
Shortness of breath
Weakness or fatigue
Swelling in your legs or abdomen
Changes in your heart rhythm
Dry or persistent cough
Skin rashes or unusual spots

Heart disease symptoms caused by valvular heart disease:

Fatigue
Shortness of breath
Irregular heartbeat
Swollen feet or ankles
Chest pain
Fainting (syncope)

1.2 Overview

The purpose for this project is create a Classification Machine Learning System using Heart Disease dataset to predict if a person have a Heart Disease or not.

Scientists have turned towards modern approaches like Data Mining and Machine Learning for predicting the disease. Machine learning proves to be effective in assisting in making decisions and predictions from the large quantity of data produced by the healthcare industry.

I will be applying Machine Learning models for classifying if a person is suffering a heart disease or not, using one of the most used dataset - Cleveland Heart Disease dataset from the UCI Repository(heart.csv).

You can found it here: http://archive.ics.uci.edu/ml/datasets/Heart+Disease or here https://www.kaggle.com/ronitf/heart-disease-uci/download

This database contains 76 attributes, but all published experiments refer to using a subset of 14 of them. In particular, the Cleveland(USA) database is the only one that has been used by ML researchers to this date. The “target” field refers to the presence of heart disease in the patient. There is a package that contains heart_disease dataset: funModeling, but we download our own data from Kaggle.

The content of the dataset (heart.csv) is divided in: age, sex, cp, trestbps, chol, fbs, restecg, thalach, exang, oldpeak, slope, ca, thal, target. We´ll explore them later.

The meaning of some of the column headers are not obvious, so, I need to change some column names by others more descriptive for our analisys:

age: The person’s age in years

sex: The person’s sex (1 = male, 0 = female)

cp -> **chest_pain_type**: The chest pain experienced (0 = typical angina, 1 = atypical angina, 2 = non-anginal pain, 3 = asymptomatic)

trestbps -> **resting_blood_pressure**: The person’s resting blood pressure (mmHg on admission to the hospital)

chol -> **cholesterol**: The person’s cholesterol measurement in mg/dl

fbs -> **fasting_blood_sugar**: The person's fasting blood sugar (> 120 mg/dl, 1 = yes; 0 = no)

restecg -> **rest_ecg**: Resting electrocardiographic measurement (0 = normal, 1 = having ST-T wave abnormality, 2 = showing probable or definite left ventricular hypertrophy by Estes' criteria)

thalach -> **max_heart_rate_achieved**: The person's maximum heart rate achieved

exang -> **exercise_induced_angina**: Exercise induced angina (1 = yes; 0 = no)

oldpeak -> **st_depression**: ST depression induced by exercise relative to rest ('ST' relates to positions on the ECG plot)

slope -> **st_slope**: the slope of the peak exercise ST segment (0 = upsloping, 1 = flat, 2 = downsloping)

ca -> **num_major_vessels**: The number of major vessels (0-4)

thal -> **thallium_stress_test**: Cold spots on the scan, where no thallium shows up, indicate areas of the heart that are not getting an adequate supply of blood (0 = non, 1 = normal, 2 = fixed defect, 3 = reversable defect)

target -> **disease**: Heart disease (1 = yes, 0 = no)

Steps to follow. . . ..

We will download the heart.csv data and run code to generate our datasets, splitting it: **edx**(80% of the data) and **validation**(20% of the data). We choose 80/20 because the number of observations (303 rows) is too little, so we need more validation items.

Next we have to clean and transform our data to permit us a clear and efficient process and apply our machine learning algorithms to obtain the most precise one predicting Heart Disease(highest accuracy).

The optimal model will be selected based on its accuracy, sensitivity, amongst other factors. This is obtained with confusionMatrix() function.

A general improvement to using overall accuracy is to study sensitivity and specificity separately. Sensitivity, also known as the true positive rate or recall(TP), is the proportion of actual positive outcomes correctly identified as such. Specificity, also known as the true negative rate(TN), is the proportion of actual negative outcomes that are correctly identified as such.

A Confusion Matrix tabulates each combination of prediction and actual value. We can create a confusion matrix in R using the table function or the confusionMatrix() function from the caret package.

To provide precise definitions, we name the four entries of the confusion matrix:

	Actually Positive	Actually Negative
Predicted positive	True positives (TP)	False positives (FP)
Predicted negative	False negatives (FN)	True negatives (TN)

The multiple names can be confusing, so here is a table to remember the terms. The table includes a column that shows the definition if we think of the proportions as probabilities.

Measure of	Name 1	Name 2	Definition	Probability representation
sensitivity	TPR	Recall	$\frac{TP}{TP+FN}$	$Pr(\hat{Y} = 1 \mid Y = 1)$
specificity	TNR	1-FPR	$\frac{TN}{TN+FP}$	$Pr(\hat{Y} = 0 \mid Y = 0)$
specificity	PPV	Precision	$\frac{TP}{TP+FP}$	$Pr(Y = 1 \mid \hat{Y} = 1)$

The **TPR** is True Positive Rate, **FPR** is False Positive Rate, and **PPV** is Positive Predictive Value. The caret function **confusionMatrix()** computes all these metrics for us once we define what category "positive"

is. The function expects factors as input, and the first level is considered the positive outcome or Y = 1.
The general approach to defining “best” in machine learning modeling is to define a **loss function**, the root mean squared error (RMSE), which can be applied to both categorical and continuous data.
If the outcomes are binary(this is the case: heart disease = yes or no), **RMSE is equivalent to one minus accuracy**, since

$$(y.hat - y)^2$$

is **0** if the prediction was correct and **1** otherwise. In general, our goal is to **build an algorithm that minimizes the loss** so it is as close **to 0** as possible.
In this report we´ll do data exploration, visualization, pre-processing, evaluate machine learning algorithms and confusionMatrix analysis to create the best predictive model.

1.3 Executive Summary

After a initial data cleaning, pre-processing and exploration, **the prediction system builted on the train dataset are evaluated on the test dataset (edx splitted) and the best model is chosen based on the analisys of confusionMatrix values and evaluate using original edx and validation datasets**.
This is to avoid overfitting.
We´ll use different models, from the simplest possible prediction system to more complex ones.

2. Methods and Analysis

We install R packages if they don´t exist, download and read “heart.csv” file, process it, ...

2.1 Data exploration and visualization

The **heart dataset** is a data table made of 14 variables (columns) and a total of 303 observations (rows).

How are the first rows of Hearts dataset? Each row represents a person with heart disease or not. . .

```
## # A tibble: 303 x 14
##          age           sex chest_pain_type resting_blood_p~ cholesterol
##        <int> <int>                 <int>              <int>       <int>
##    1      63     1                     3                145         233
##    2      37     1                     2                130         250
##    3      41     0                     1                130         204
##    4      56     1                     1                120         236
##    5      57     0                     0                120         354
##    6      57     1                     0                140         192
##    7      56     0                     1                140         294
##    8      44     1                     1                120         263
##    9      52     1                     2                172         199
##   10      57     1                     2                150         168
## # ... with 293 more rows, and 9 more variables: fasting_blood_sugar <int>,
## #   rest_ecg <int>, max_heart_rate_achieved <int>,
## #   exercise_induced_angina <int>, st_depression <dbl>, st_slope <int>,
## #   num_major_vessels <int>, thallium_stress_test <int>, disease <int>
```

We can see the number of NA into the dataset:

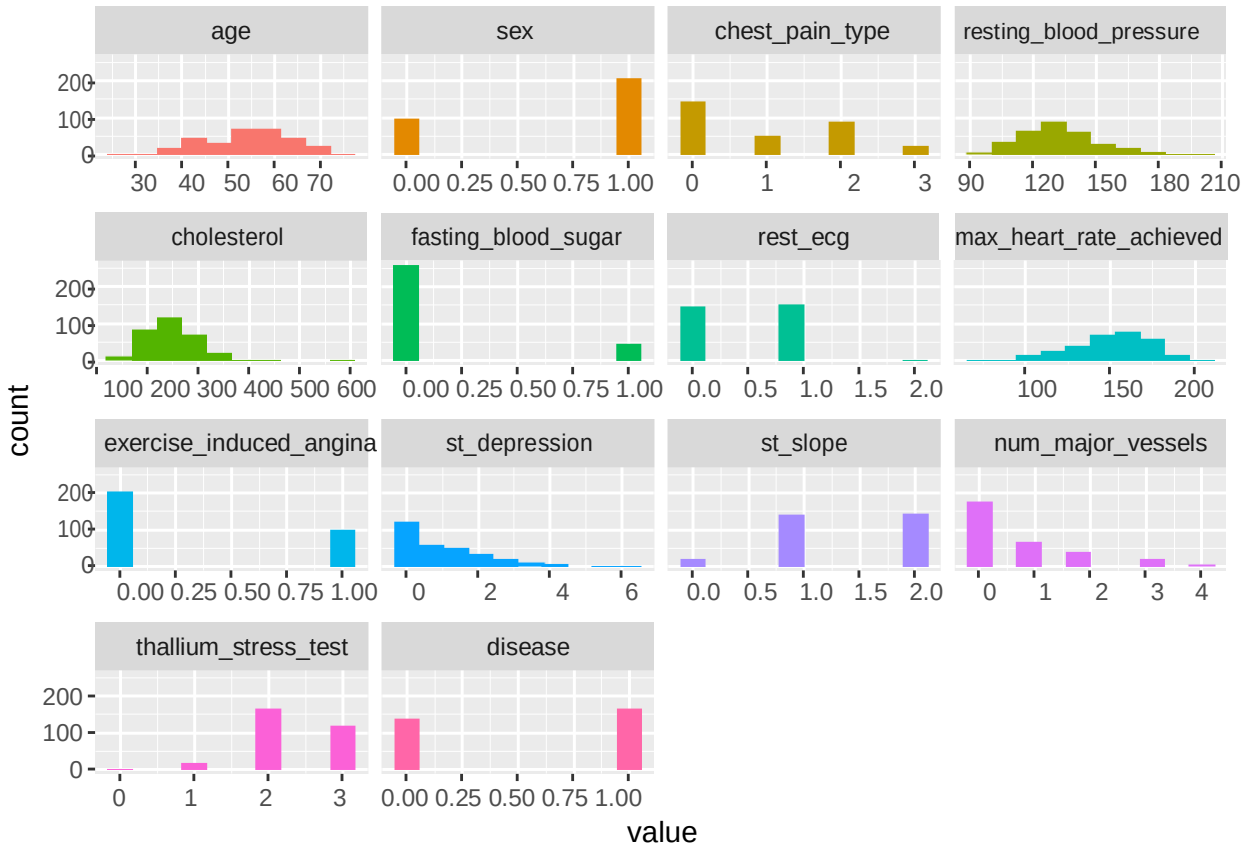
```
##                       age                       sex         chest_pain_type
##                         0                         0                       0
##  resting_blood_pressure               cholesterol       fasting_blood_sugar
##                         0                         0                       0
##                  rest_ecg   max_heart_rate_achieved   exercise_induced_angina
```

```
##                0                0                0
##   st_depression         st_slope num_major_vessels
##                0                0                0
##  thallium_stress_test          disease
##                0                0
```

There are no missing values (NA).

Now, let´s see a brief summary and plot distribution for each variable:

```
##          age              sex         chest_pain_type resting_blood_pressure
##  Min.   :29.00   Min.   :0.0000   Min.   :0.000   Min.   : 94.0
##  1st Qu.:47.50   1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:120.0
##  Median :55.00   Median :1.0000   Median :1.000   Median :130.0
##  Mean   :54.37   Mean   :0.6832   Mean   :0.967   Mean   :131.6
##  3rd Qu.:61.00   3rd Qu.:1.0000   3rd Qu.:2.000   3rd Qu.:140.0
##  Max.   :77.00   Max.   :1.0000   Max.   :3.000   Max.   :200.0
##     cholesterol     fasting_blood_sugar    rest_ecg
##  Min.   :126.0   Min.   :0.0000      Min.   :0.0000
##  1st Qu.:211.0   1st Qu.:0.0000      1st Qu.:0.0000
##  Median :240.0   Median :0.0000      Median :1.0000
##  Mean   :246.3   Mean   :0.1485      Mean   :0.5281
##  3rd Qu.:274.5   3rd Qu.:0.0000      3rd Qu.:1.0000
##  Max.   :564.0   Max.   :1.0000      Max.   :2.0000
##  max_heart_rate_achieved exercise_induced_angina st_depression
##  Min.   : 71.0           Min.   :0.0000          Min.   :0.00
##  1st Qu.:133.5           1st Qu.:0.0000          1st Qu.:0.00
##  Median :153.0           Median :0.0000          Median :0.80
##  Mean   :149.6           Mean   :0.3267          Mean   :1.04
##  3rd Qu.:166.0           3rd Qu.:1.0000          3rd Qu.:1.60
##  Max.   :202.0           Max.   :1.0000          Max.   :6.20
##     st_slope     num_major_vessels thallium_stress_test      disease
##  Min.   :0.000   Min.   :0.0000    Min.   :0.000         Min.   :0.0000
##  1st Qu.:1.000   1st Qu.:0.0000    1st Qu.:2.000         1st Qu.:0.0000
##  Median :1.000   Median :0.0000    Median :2.000         Median :1.0000
##  Mean   :1.399   Mean   :0.7294    Mean   :2.314         Mean   :0.5446
##  3rd Qu.:2.000   3rd Qu.:1.0000    3rd Qu.:3.000         3rd Qu.:1.0000
##  Max.   :2.000   Max.   :4.0000    Max.   :3.000         Max.   :1.0000
```

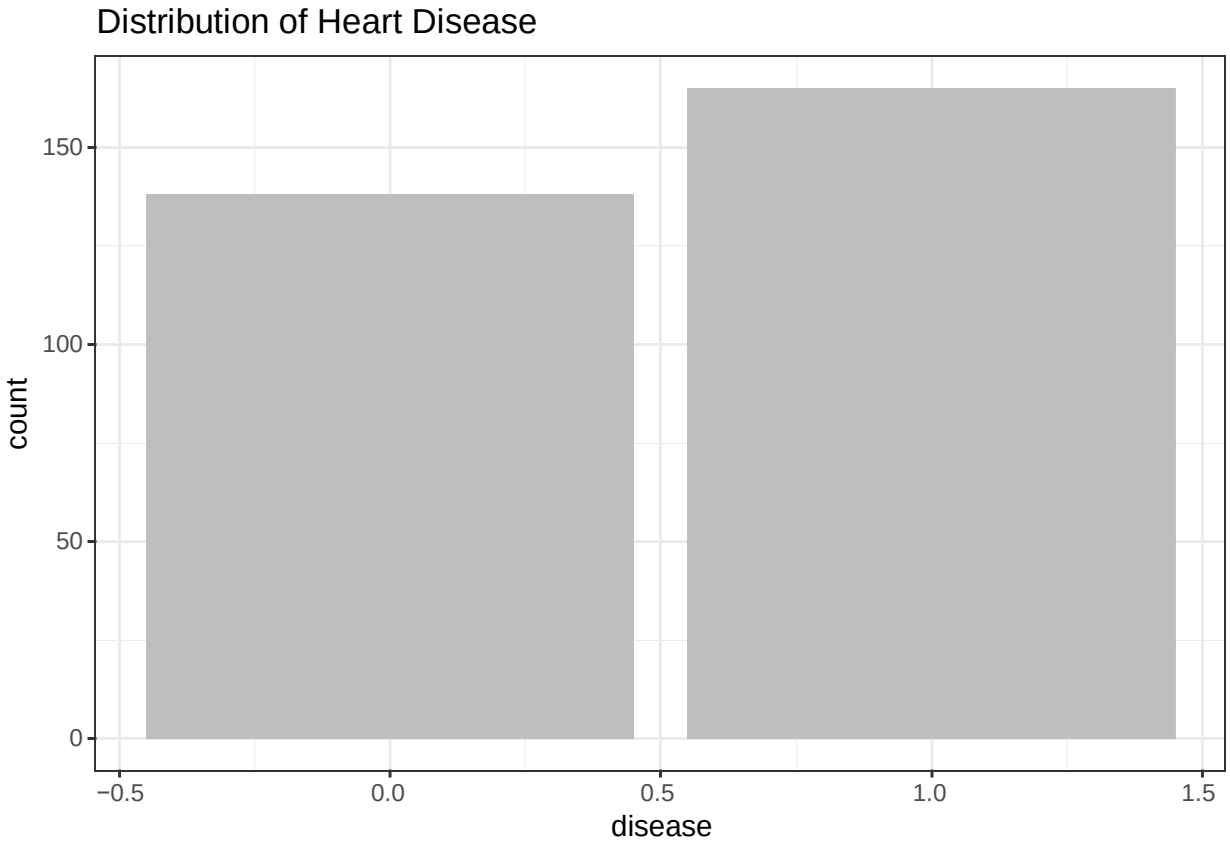


Above, we can see several variables with normal distribution, others are binary categorical (0, 1), and the rest with several categorical levels.

Let´s see the proportion and distribution of “disease”(heart disease) variable:

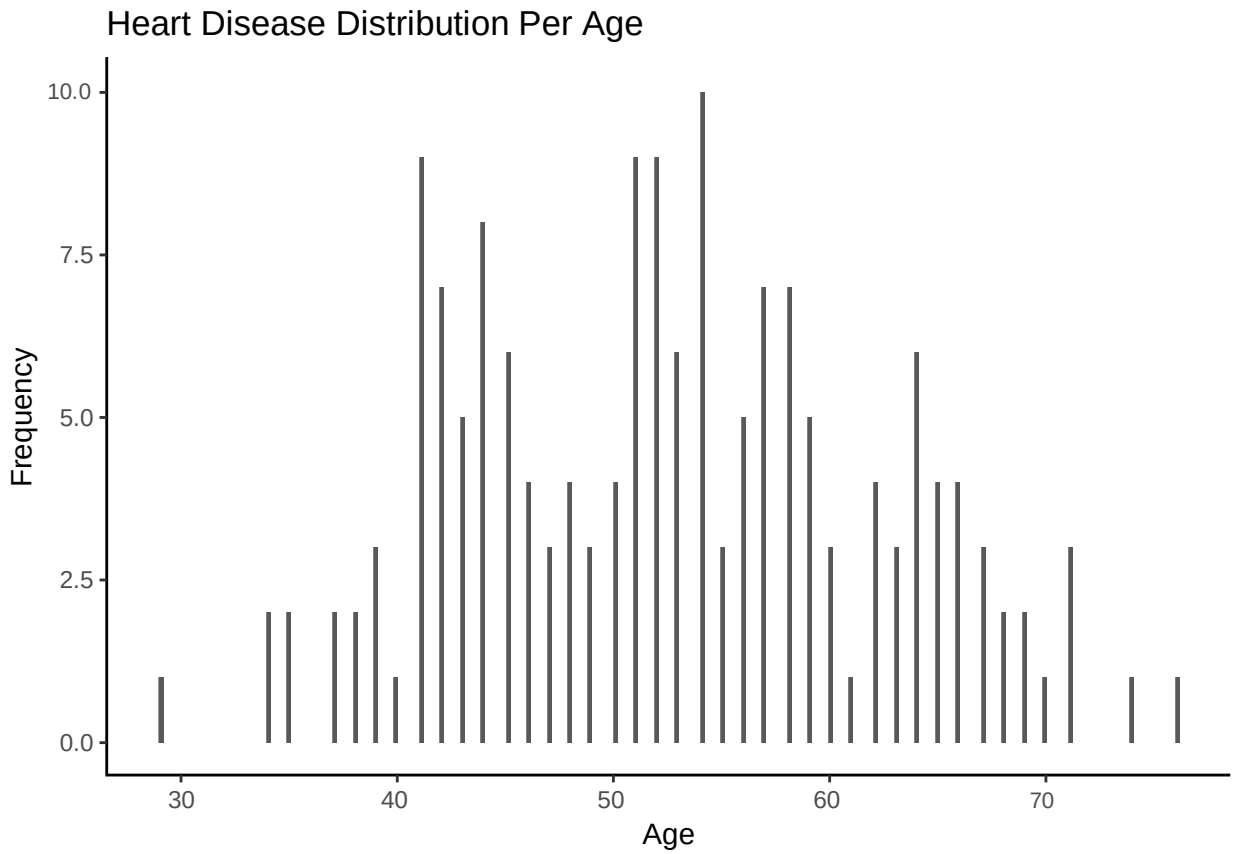
Note: “0” = No disease , “1” = Have disease

```
##
##    0    1
## 0.46 0.54
```

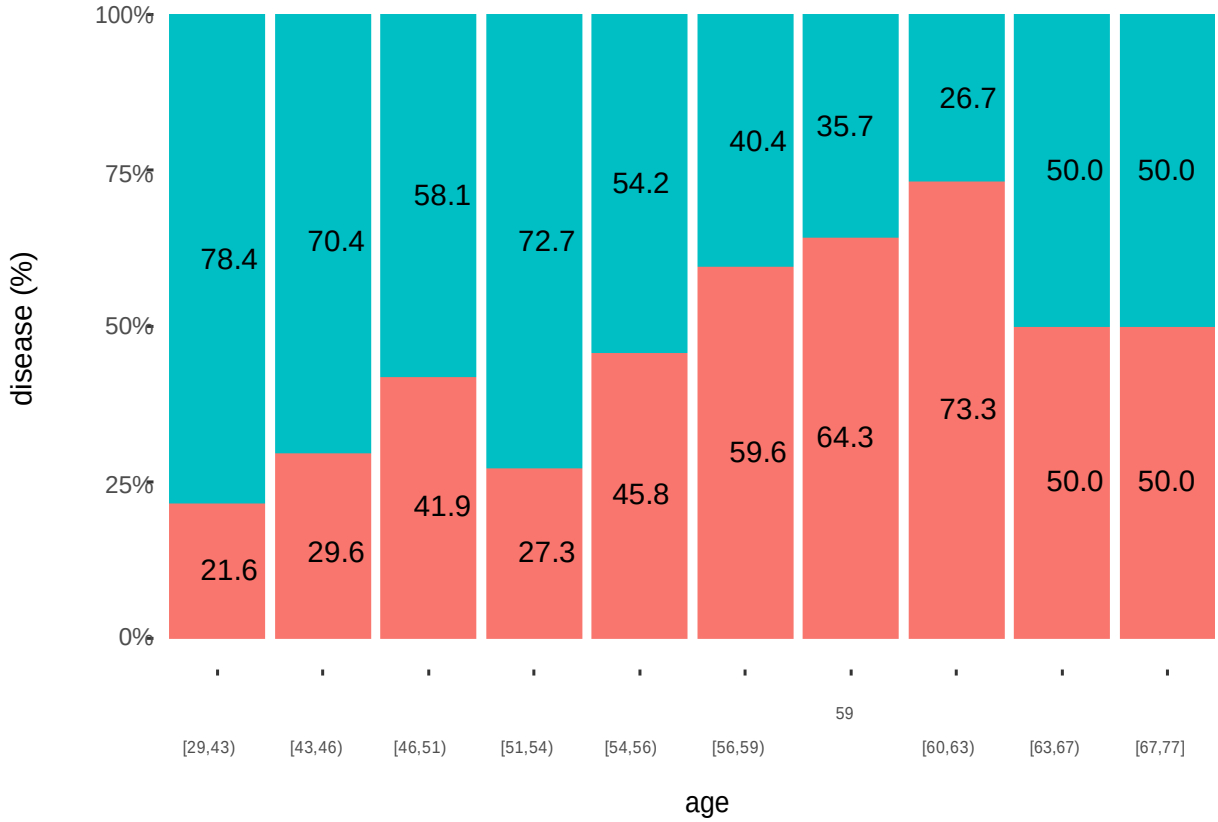


This means that there is NO heart disease present in 46% of patients and the remaining 54% HAVE heart disease.

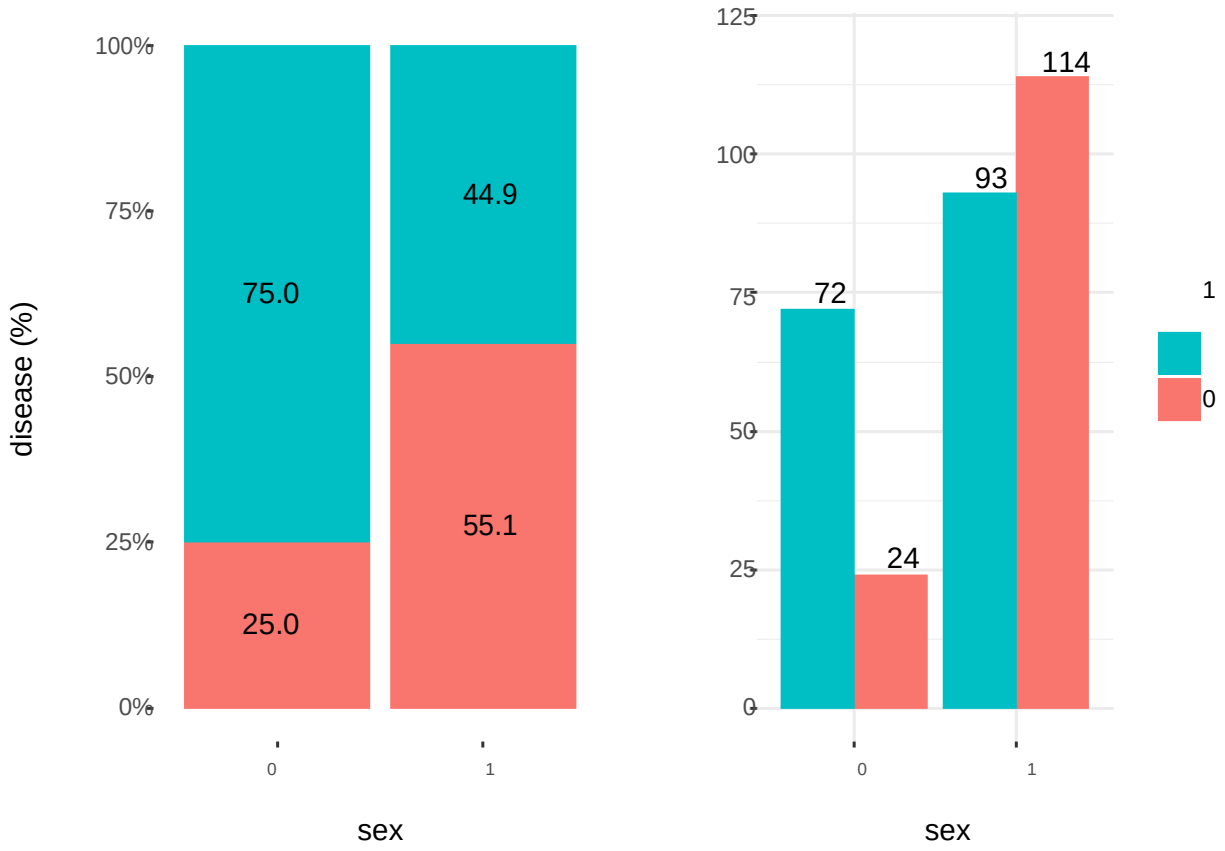
Next, we see the distribution of Heart Disease per Age(quantity):



Above, we see that people with 54 years old have more heart disease, and one step lower with 41 year old.

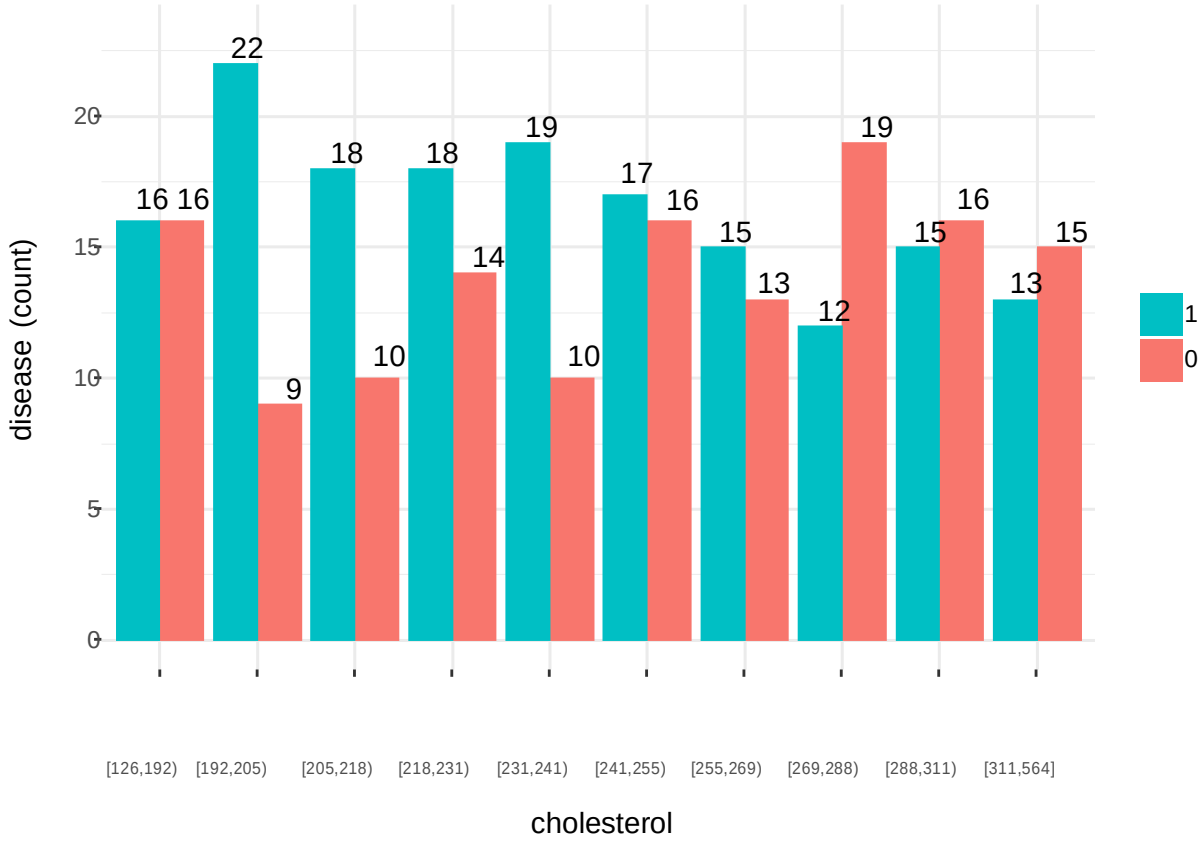


Above, we can see another type of distribution on this dataset of Heart Disease per Age(percentual, cyan = heart disease, red = no heart disease).

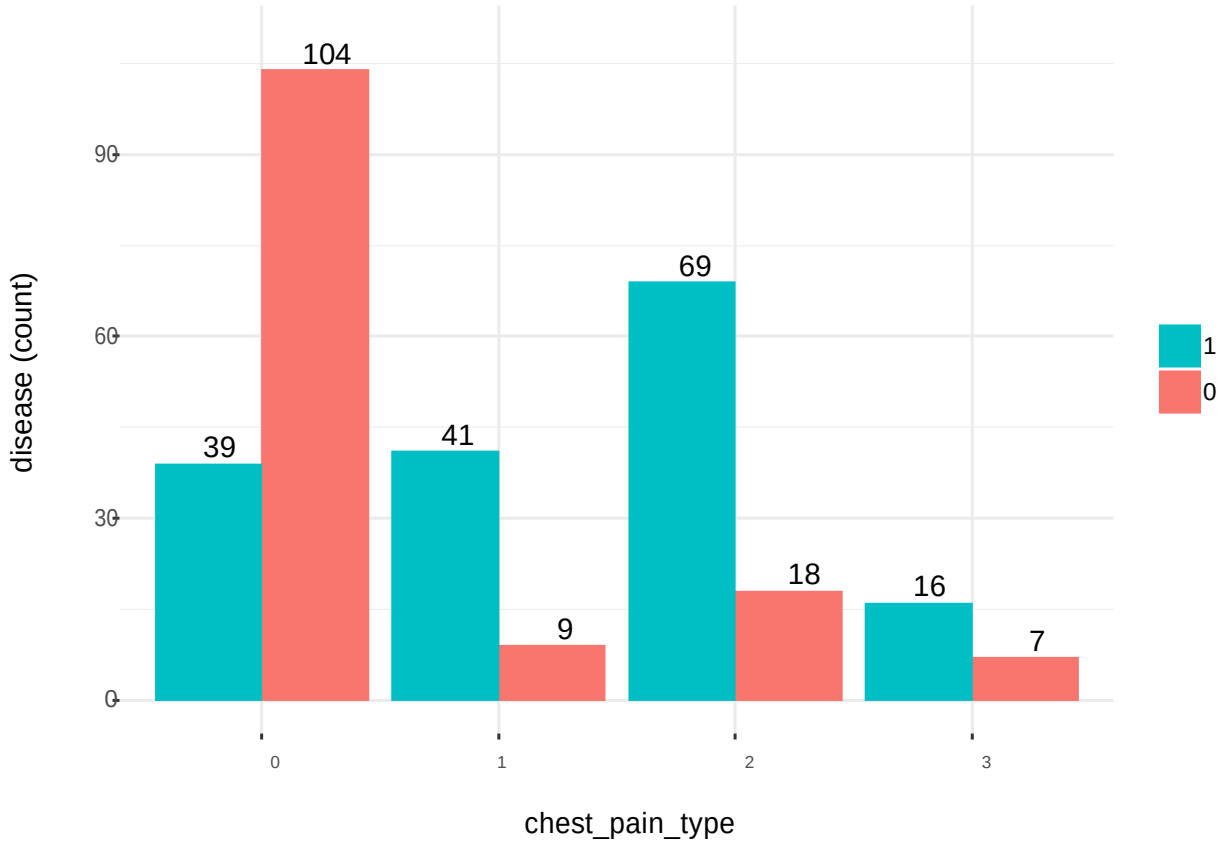


Above, we can see another type of distribution on this dataset of Heart Disease per Sex/Disease: cyan = heart disease, red = no heart disease; sex: 0 = Female, 1 = Male).

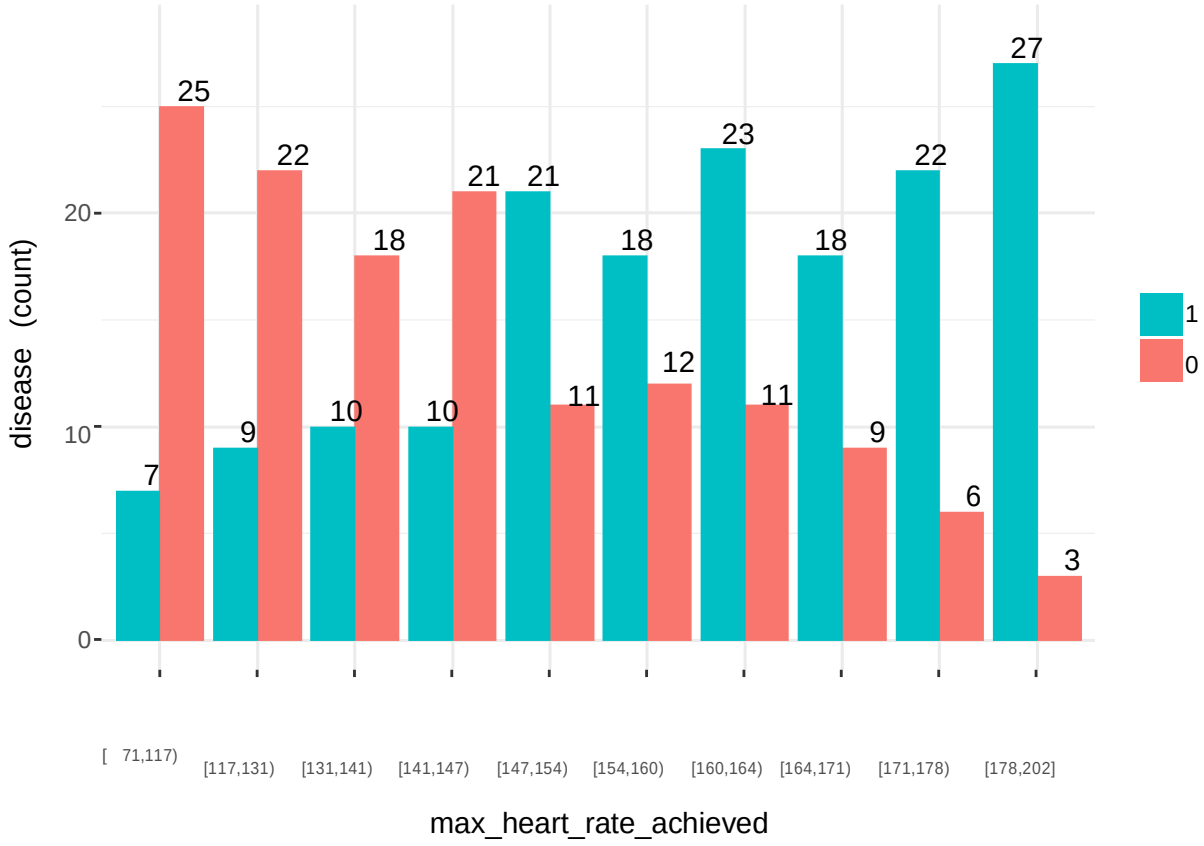
Note that total Female are 96 and total Male are 207, this is not balanced.



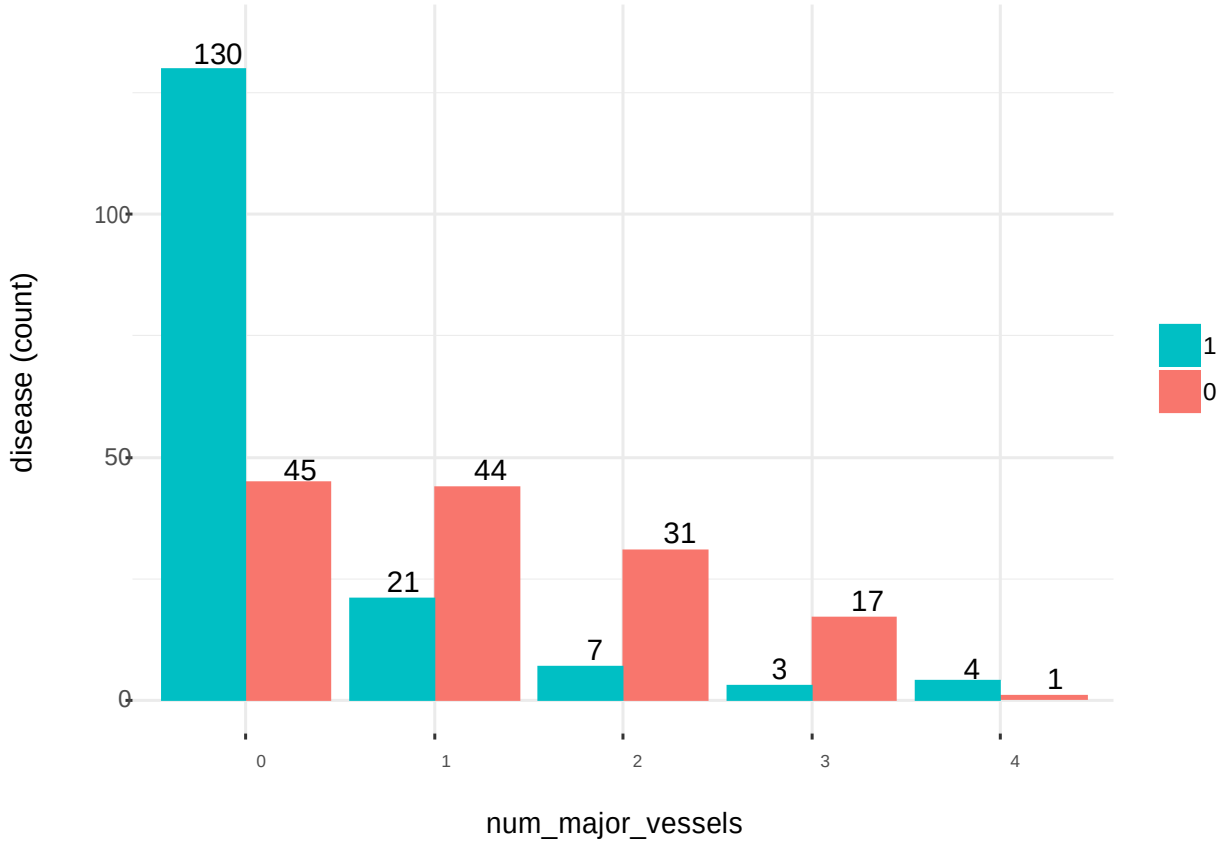
Above, we can see distribution Cholesterol vs Disease: cyan = heart disease, red = no heart disease).



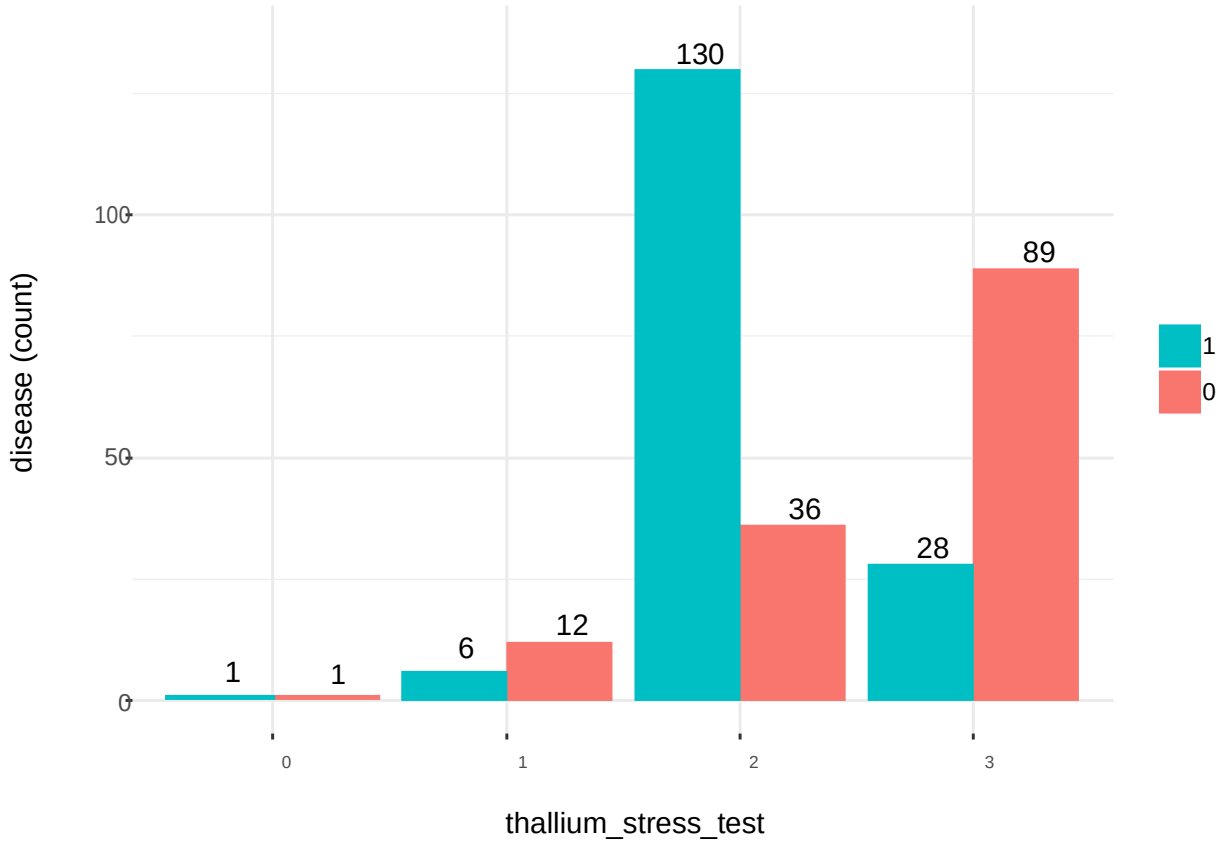
Above, we can see distribution chest_pain_type vs Disease: cyan = heart disease, red = no heart disease; chest_pain_type : 0 = typical angina, 1 = atypical angina, 2 = non-anginal pain, 3 = asymptomatic.



Above, we can see distribution cmax_heart_rate_achieved vs Disease: cyan = heart disease, red = no heart disease. We detect that to higher heart rate, more cases of heart disease.



Above, we can see distribution num_major_vessels vs Disease: cyan = heart disease, red = no heart disease; num_major_vessels = 0 to 4. We can see that as the number of major blood vessels increases, the probability of heart disease decreases. That have sense, as it means more blood can get to the heart.

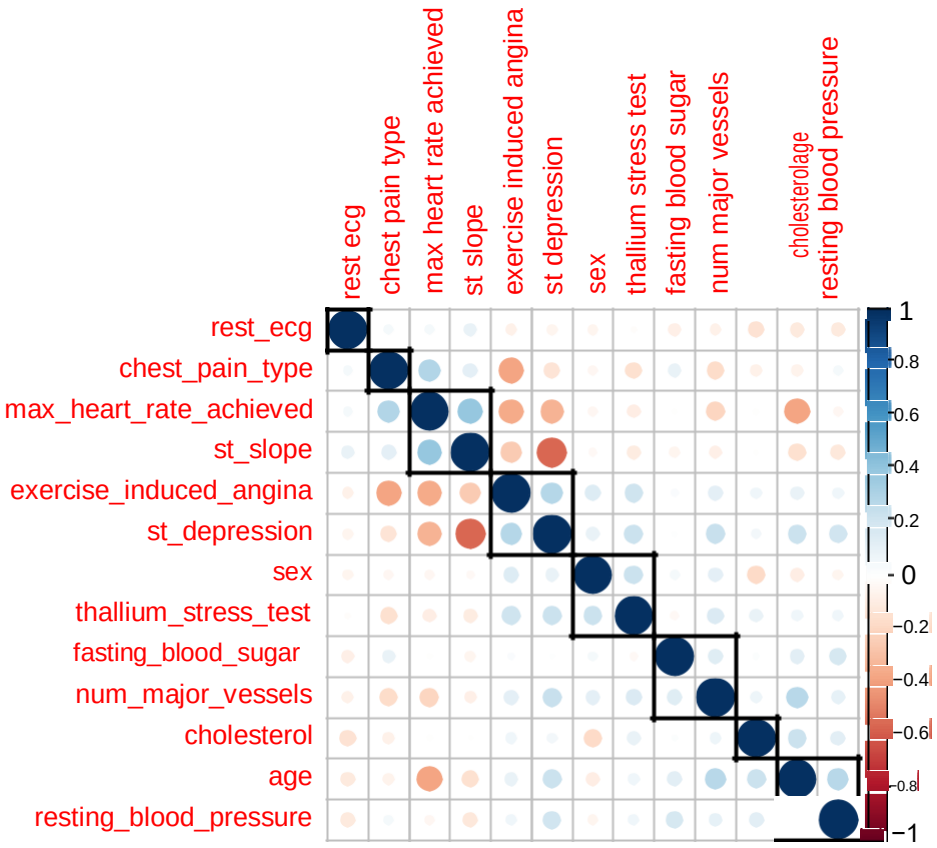


Above, we can see distribution thallium_stress_test vs Disease: cyan = heart disease, red = no heart disease; thallium_stress_test : 0 = non, 1 = normal, 2 = fixed defect, 3 = reversable defect. We can see that with a thallium_stress_test = 2 = fixed defect, a heart disease have a explosion. Cold spots on the scan, where no thallium shows up, indicate areas of the heart that are not getting an adequate supply of blood. This is the most important variable of our dataset “heart.csv”.

Let’s keep going. . . .

Most machine learning algorithms assume that the predictor variables are independent from each others. Let´s see correlations between predictors, if we have values near to 1, we have to discard that variable:

	Variable	disease
1	disease	1.00
2	chest_pain_type	0.43
3	max_heart_rate_achieved	0.42
4	st_slope	0.35
5	rest_ecg	0.14
6	cholesterol	-0.09
7	resting_blood_pressure	-0.14
8	age	-0.23
9	thallium_stress_test	-0.34
10	num_major_vessels	-0.39
11	st_depression	-0.43



As we see, there are NO correlations between variables(predictors). Always < 0.43, so we are ok to procces and modeling our data. Just we have to separate “disease” variable(the outcome).

2.2 Data pre-processing

Now we´ll continue with data pre-processing. . . .From here, we will show the code for better understanding. **Principal Component Analysis(PCA)**

We can get variable importance without using a predictive model using information theory, ordered from highest to lowest:

Note:

en: entropy measured in bits

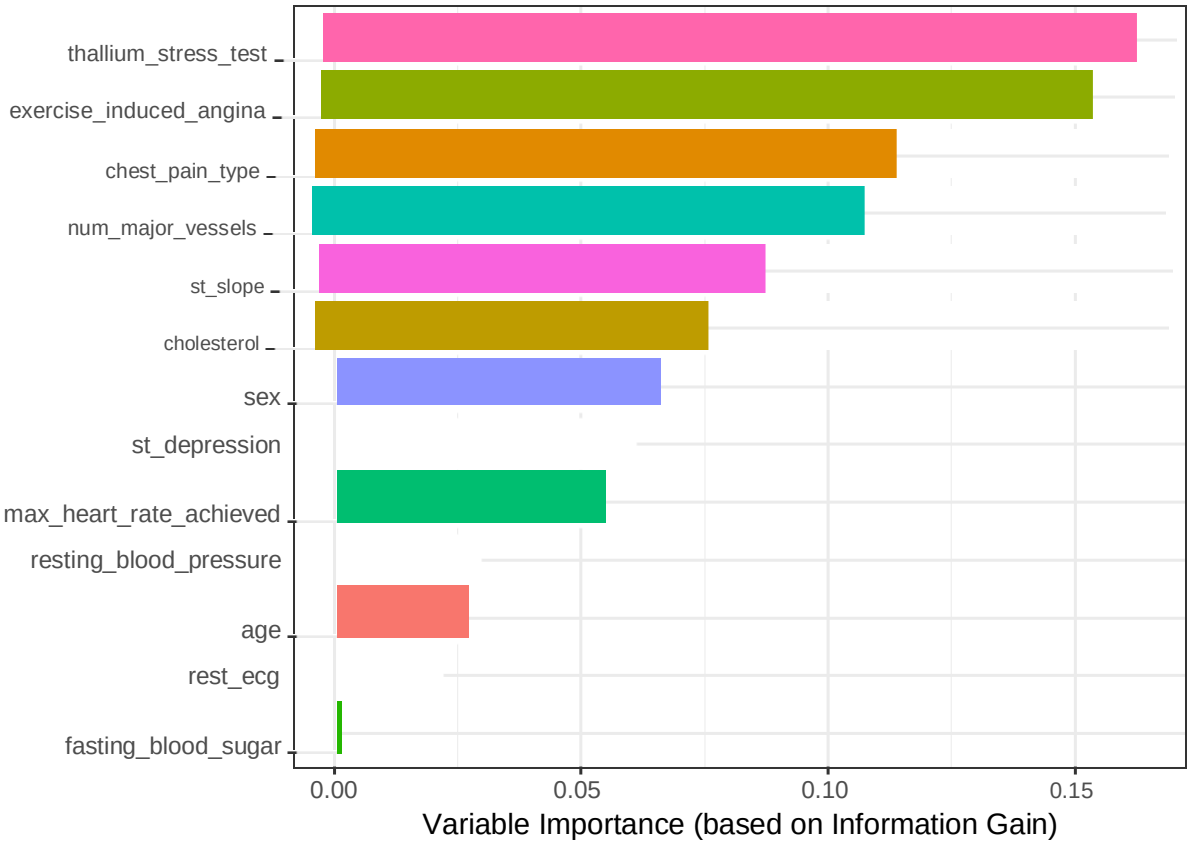
mi: mutual information

ig: information gain

gr: gain ratio (is the most important metric here, ranged from 0 to 1, with higher being better)

```
##                                 var    en    mi                ig                gr
##  1thallium_stress_test 2.077 0.213 0.2132507020 0.1646099129
##  2  exercise_induced_angina 1.764 0.142 0.1422104940 0.1560089774
##  3chest_pain_type 2.529 0.205 0.2045988556 0.1176236333
##  4num_major_vessels 2.474 0.186 0.1859573080 0.1116193825
## 5                        st_slope 2.171 0.117 0.1168337959 0.0902894816
```

```
 ## 6                    cholesterol 7.478  0.560 0.5597365269 0.0794675650
 ##  7                            sex 1.836  0.059 0.0591383086 0.0656433918
 ##  8                 st_depression  4.869  0.253 0.2533884114 0.0613803042
## 9  max_heart_rate_achieved 6.828 0.335 0.3350981472 0.0543227461
## 10  resting_blood_pressure 5.551 0.140 0.1404280402 0.0298947640
 ## 11                           age 5.937  0.135 0.1353052791 0.0266477255
 ##  12                      rest_ecg  2.058  0.024 0.0240748363 0.0221288889
 ##  13          fasting_blood_sugar  1.600  0.001 0.0005658824 0.0009336281
```



According to the plot, “thallium_stress_test” is the most important variable in this dataset.

Now, we transform numeric “disease” variable as a factor, because confusionMatrix() function needs work with factors of the same length,

```
# converts disease as factor "0" or "1" to use with confusionMatrix()

Hearts <- mutate_at(Hearts, vars(disease), as.factor)

str(Hearts$disease)
```

```
##    Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
```

Then, “disease” column is removed, then we scale and center the data. Compute PCA,

```
# now calculate PCA removing "disease" with correlation "1"

set.seed(1)
pca <- prcomp(Hearts %>% select(-disease), scale = TRUE, center = TRUE)
```

```
str(pca)
```

```
## List of 5
##  $ sdev     : num [1:13] 1.66 1.24 1.11 1.09 1.01 ...
##  $ rotation: num [1:13, 1:13] -0.3142 -0.0908 0.2746 -0.1839 -0.1174 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:13] "age" "sex" "chest_pain_type" "resting_blood_pressure" ...
##   .. ..$ : chr [1:13] "PC1" "PC2" "PC3" "PC4" ...
##  $ center  : Named num [1:13] 54.366 0.683 0.967 131.624 246.264 ...
##   ..- attr(*, "names")= chr [1:13] "age" "sex" "chest_pain_type" "resting_blood_pressure" ...
##  $ scale    : Named num [1:13] 9.082 0.466 1.032 17.538 51.831 ...
##   ..- attr(*, "names")= chr [1:13] "age" "sex" "chest_pain_type" "resting_blood_pressure" ...
##  $ x: num [1:303, 1:13] -0.623 0.455 1.826 1.713 0.371 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:13] "PC1" "PC2" "PC3" "PC4" ...
##  - attr(*, "class")= chr "prcomp"
```

```
summary(pca)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6
## Standard deviation     1.6622 1.2396 1.10582 1.08681 1.01092 0.98489
## Proportion of Variance 0.2125 0.1182 0.09406 0.09086 0.07861 0.07462
## Cumulative Proportion  0.2125 0.3307 0.42481 0.51567 0.59428 0.66890
##                           PC7     PC8    PC9    PC10    PC11    PC12
## Standard deviation     0.92885 0.88088 0.8479 0.78840 0.72808 0.65049
## Proportion of Variance 0.06637 0.05969 0.0553 0.04781 0.04078 0.03255
## Cumulative Proportion  0.73527 0.79495 0.8503 0.89807 0.93885 0.97140
##                           PC13
## Standard deviation     0.6098
## Proportion of Variance 0.0286
## Cumulative Proportion  1.0000
```

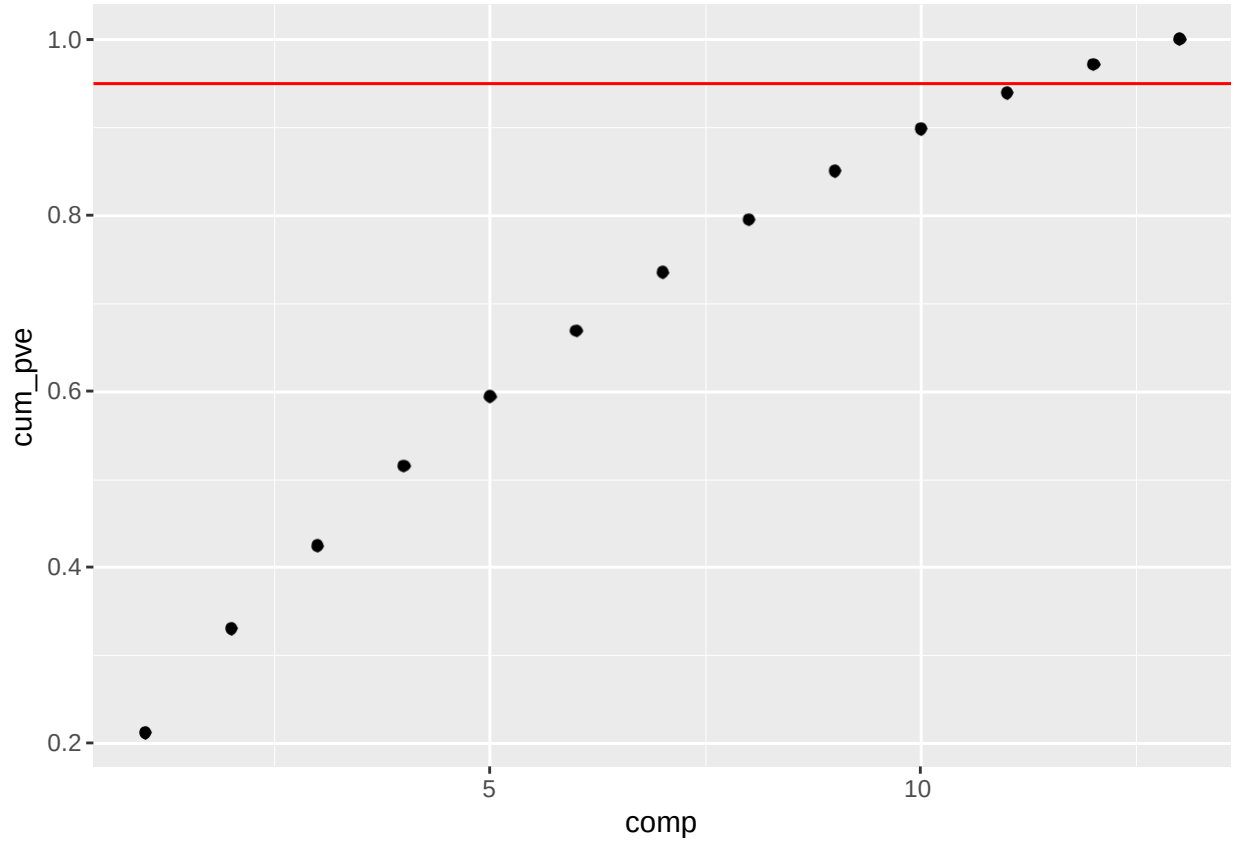
As we see, standard deviation(sdev) of each column is stored in pca$sdev, so the total variability in our data can be defined as the sum of the sum of squares of the columns. We assume the columns are centered, so this sum is equivalent to the sum of the variances of each column:

```
# Compute the proportion of variance explained(PVE)

pve_Hearts <- pca$sdev^2 / sum(pca$sdev^2)
cum_pve <- cumsum(pve_Hearts)       # Cummulative percent explained
pve_table <- tibble(comp = seq(1:ncol(Hearts %>% select(-disease))),
                                pve_Hearts, cum_pve)

#  plot components vs PVE to see if there are correlations
#  to reduce variable numbers. All have to be independent variables

ggplot(pve_table, aes(x = comp, y = cum_pve)) +
   geom_point() +
   geom_abline(intercept = 0.95, color = "red", slope = 0) #
                        line at 95% of cum_pve
```



As we see, the above plot shows that 95% of the variance is explained with almost all principal components(11 PCs), so we decide to work with the originals 13 variables dataset.

And now, we create edx and validation datasets:

```
#############################################################
#    Spliting Hearts dataset in edx and validation sets
#    We will use it in the final algorithm
#############################################################

set.seed(1) # if you are using R 3.5 or Microsoft R Open 3.5.3
#  set.seed(1, sample.kind="Rounding") if using R 3.5.3 or later

#  Validation set will be 20% of Hearts data because it is a little dataset

test_index <- createDataPartition(y = Hearts$disease,
                                  times = 1, p = 0.2, list = FALSE)
edx <- Hearts[-test_index,]
validation <- Hearts[test_index,] # we will use it only to do final test
```

We will split edx data into train_set and test_set. Validation data will be used only to test the final value of the best model, this is to avoid overfitting (too optimistic result):

```
#############################################################
#    Spliting edx dataset in train_set and test_set
#    We will use it to train ours models
#############################################################
```

```
set.seed(1) # if you are using R 3.5 or Microsoft R open 3.5.3
# set.seed(1, sample.kind="Rounding") # if using R 3.5.3 or later

test_index <- createDataPartition(y = edx$disease, times =
                                    1, p = 0.2,
                                    list = FALSE)      # test_set 20%
train_set <- edx[-test_index,]
test_set <- edx[test_index,]
```

2.3 Model building approach

As we see before, we´ll use a heart.csv dataset preprocessed, with 303 observations and 13 variables to construct and test our prediction of heart disease models.

First, we´ll construct a function to apply several models, with train and test sets. Next we´ll do it with edx and validation datasets. With this, we can compute accuracy average of each model and choose the one with the smallest difference between its values with differents datasets, reducing the variation of prediction. So, in theory, if we get a larger dataset of heart disease, there will be a little difference in the prediction result, within 95% of the confidence interval.

To start, we choose these models of Classification: (you can see a list of models here: https://topepo.github. io/caret/available-models.html)

glm: Generalized Linear Model

lda: Linear Discriminant Analysis

naive_bayes: Naive Bayes

svmLinear: Support Vector Machines with Linear Kernel

gamLoess: Generalized Additive Model using LOESS

qda: Quadratic Discriminant Analysis

knn: k-Nearest Neighbors

kknn: Other k-Nearest Neighbors

gam: Generalized Additive Model using Splines

rf: Random Forest

ranger: Other Random Forest

wsrf: Weighted Subspace Random Forest

mlp: Multi-Layer Perceptron Neural Network

```
#  model list, it can be any model you want to test,
#  just be sure to load the library that contains it

models <- c("glm", "lda", "naive_bayes", "svmLinear",
            "gamLoess", "qda", "knn", "kknn", "gam", "rf",
            "ranger", "wsrf", "mlp")

# trainControl function for control tuning parameters models

control <- trainControl(method = "cv",             # cross validation
                        number = 10,        # 10 k-folds or number
                                    #  of resampling iterations
                        repeats = 5)
```

Here are the results of differents models on 2 differents datasets: first train/test_set, and edx/validation later:

```
# initializing variables

data_train <- train_set                  # first value for data parameter
data_test <- test_set # first we´ll use train and test dataset true_value <-
test_set$disease # true outcome from test_set

# loop to use train and test set first and edx and validation later

for(i in 1:2) {
   fits <- lapply(models, function(model){
#       print(model)  # it´s used to debug code
       set.seed(1)
       train(disease ~ .,
                 method = model,
                 preProcess=c("center", "scale"),          # to normalize the data
                 data = data_train,
                 trControl = control)
   })

   names(fits) <- models

   #  to be sure that the actual value of the output
   #  has not influence on the prediction

   vali2 <- data_test %>% select(-disease)

   pred <- sapply(fits, function(object) # predicting outcome
         predict(object, newdata = vali2))

   # avg predicted values if equals to true values

   if (i == 1) acc <- colMeans(pred == true_value)

   data_train <- edx                          #      last value for data parameter
   data_test <-  validation                       # last we´ll use edx and validation
   true_value <- validation$disease           # true outcome from validation set

}

acc       # all accuracy values with first dataset. Train and Test set
```

```
##         glm         lda naive_bayes   svmLinear   gamLoess         qda
##   0.7551020   0.7346939   0.8163265   0.7959184   0.7959184   0.7551020
##         knn        kknn         gam          rf      ranger        wsrf
##   0.8163265   0.7551020   0.7755102   0.8571429   0.8163265   0.8163265
##         mlp
##   0.7755102
```

```
acc2 <- colMeans(pred == true_value) # avg predicted values

acc2     # all accuracy values with last dataset. Edx and Validation set
```

```
##           glm           lda naive_bayes     svmLinear     gamLoess           qda
##     0.9016393     0.9180328     0.8688525     0.9016393     0.9344262     0.8196721
##           knn          kknn           gam            rf        ranger          wsrf
##     0.9180328     0.9344262     0.8852459     0.9016393     0.8524590     0.8688525
##           mlp
##     0.9180328
```

Then we make the difference between the most and the least optimistic models. Here we choose the model with the smallest value(variation between datasets). It´s **Ranger Model**, other Random Forest model:

```
results <- acc2 - acc # accuracy diff by model

# show results of difference on the same model for different dataset

results
```

```
##           glm           lda naive_bayes     svmLinear     gamLoess           qda
##    0.14653730    0.18333891    0.05252593    0.10572098    0.13850786    0.06457009
##           knn          kknn           gam            rf        ranger          wsrf
##    0.10170626    0.17932419    0.10973570    0.04449649    0.03613249    0.05252593
##           mlp
##    0.14252258
```

We compute balanced accuracy, sensitivity, specificity, prevalence with help of confusionMatrix() function:

Here with edx/validation datasets,

```
# Compute balance accuracy, sensitivity, specificity,
# prevalence with confusionMatrix

cm_validation_hearts<- confusionMatrix(as.factor(pred[,11]),
                                       validation$disease, positive = "1")
cm_validation_hearts
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          025 6
##          1 327
##
##                Accuracy : 0.8525
##                  95% CI : (0.7383, 0.9302)
##     No Information Rate : 0.541
##     P-Value [Acc > NIR] : 2.6e-07
##
##                   Kappa : 0.7053
##
##  Mcnemar's Test P-Value : 0.505
##
##             Sensitivity       : 0.8182
##             Specificity       : 0.8929
##          Pos Pred Value : 0.9000
##          Neg Pred Value : 0.8065
##              Prevalence : 0.5410
##          Detection Rate : 0.4426
```

```
##       Detection Prevalence : 0.4918
##          Balanced Accuracy : 0.8555
##
##           'Positive' Class : 1
##
```

Here we see that **Prevalence = 0.54** (is the proportion of individuals with disease) is a good value (balanced). According to the American Heart Association's Heart and Stroke Statistics, 2019 Update, nearly half (48 percent, 121.5 million in 2016) of all adults in the United States have some type of cardiovascular disease(https://www.sciencedaily.com/releases/2019/01/190131084238.htm).

Sensitivity = 0.81 (also known as the true positive rate or recall, is the proportion of actual positive outcomes correctly identified as such),

Specificity = 0.89 (also known as the true negative rate, is the proportion of actual negative outcomes that are correctly identified as such), **Detection Rate = 0.44** and **Balanced Accuracy = 0.85** is very good.

2.3.1 Inside Ranger Method: a Random Forest Model

Ranger method is a fast implementation of Random Forests. Ensembles of classification, regression, survival and probability prediction trees are supported. Data from genome-wide association studies can be analyzed efficiently. In spite of particularly suited for high dimensional data (ours is tiny now, huge in near future), we´ll use it changing several default parameters.

```
# Final Ranger model algorithm computed with principal
# edx and validation dataset

# to avoid error in confusionMatrix, we convert num 0,1 to No,Yes

levels(edx$disease) <- c("No", "Yes")
levels(validation$disease) <- c("No", "Yes")

# to be sure that the actual value of the output
# has not influence on the prediction

vali2 <- validation %>% select(-disease)

# trainControl function for control iteration model
# we test differents parameters and choose that ones that improve accuracy

control <- trainControl(method = "cv",             # cross validation
                        number = 30)     # optimum k-folds or number 30
                                         # of resampling iterations

# training Ranger Model

set.seed(1)
ranger_model <- train(disease ~., data = edx,
                      method = "ranger",      # ranger model
                      preProcess=c("center", "scale"), # to normalize the data trControl =
                      control)

# predicting outcome

prediction_ranger <- predict(ranger_model, newdata = vali2)
```

```
# Check results with confusionMatrix() function and validation set

cm_ranger_model <- confusionMatrix(prediction_ranger,
                                   validation$disease, positive = "Yes")
cm_ranger_model
```

```
## Confusion Matrix and Statistics
##
##                 Reference
## Prediction No Yes
##            No  25   3
##            Yes  3  30
##
##                Accuracy : 0.9016
##                  95% CI : (0.7981, 0.963)
##     No Information Rate : 0.541
##     P-Value [Acc > NIR] : 1.252e-09
##
##                   Kappa : 0.8019
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity       : 0.9091
##             Specificity       : 0.8929
##          Pos Pred Value : 0.9091
##          Neg Pred Value : 0.8929
##              Prevalence : 0.5410
##          Detection Rate : 0.4918
##    Detection Prevalence : 0.5410
##       Balanced Accuracy : 0.9010
##
##        'Positive' Class : Yes
##
```

Now we have a **better Balanced Accuracy = 0.90 , better Detection Rate = 0.49 and better Sensitivity = 0.90 (best to improve patient health)**, and the same Specificity = 0.85 (true negative rate, not harmful to people's health).

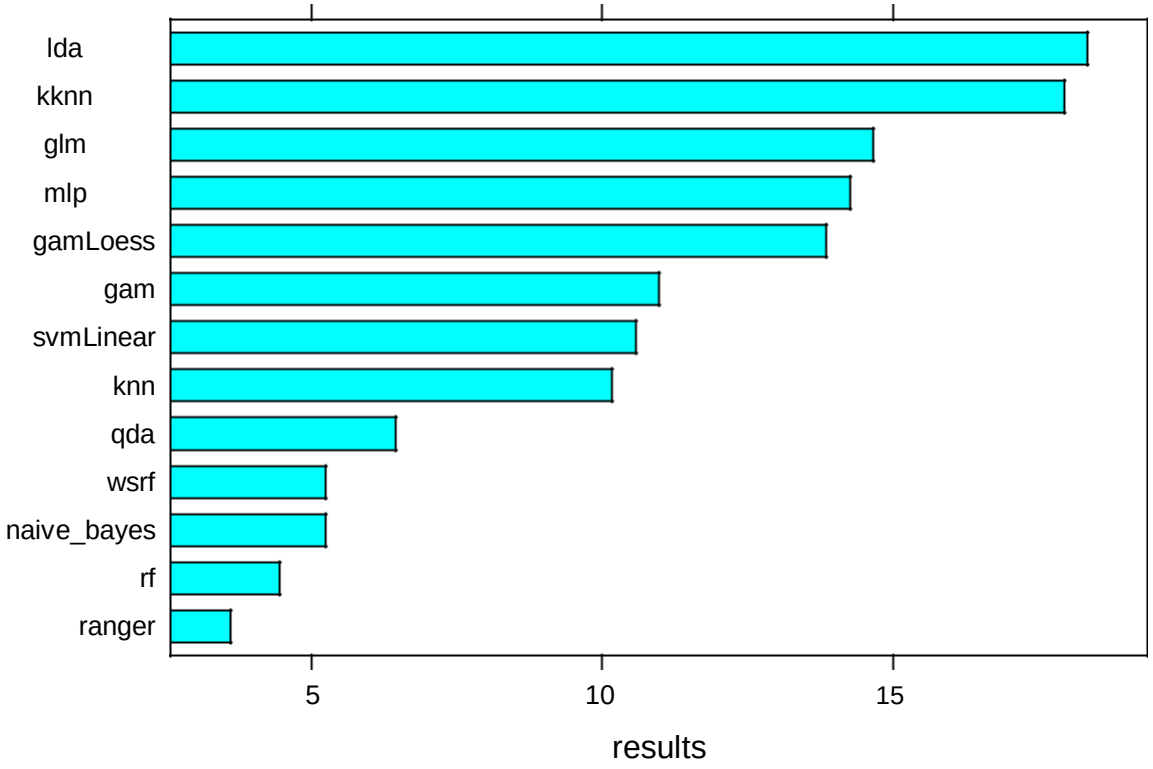
3. Results

These are the results that we were able to achieve with our models, trained with ***edx*** and tested with **validation** datasets:

	Model Accuracy
glm	0.9016393
lda	0.9180328
naive_bayes	0.8688525
svmLinear	0.9016393
gamLoess	0.9344262
qda	0.8196721
knn	0.9180328
kknn	0.9344262
gam	0.8852459
rf	0.9016393
ranger	0.8524590
wsrf	0.8688525
mlp	0.9180328

Here is the table of differences in the results of the same models tested with 2 different datasets:

	Accuracy difference by model in %
glm	14.653730
lda	18.333891
naive_bayes	5.252593
svmLinear	10.572098
gamLoess	13.850786
qda	6.457009
knn	10.170626
kknn	17.932419
gam	10.973570
rf	4.449649
ranger	3.613249
wsrf	5.252593
mlp	14.252258



In the barchart above, difference between 2 datasets on the same model, smaller is better. This means that has a better precision outcome stability when there are differents dataset as input.

And here is the Accuracy result from our **FINAL Ranger Model**:

[1] "Ranger Model Accuracy: 90.16 %"

4. Conclusion

As we see, studying one of the most used dataset — Cleveland Heart Disease dataset from the UCI Repository(heart.csv), **with our Classification Machine Learning Algorithm ("ranger" random forest model), we reach accuracy and sensitivity of 90%, that is very important because with these parameters as high as possible, more people we´ll be predicted with Heart Disease and they will be treated to improve their quality of life**. Other models like "kknn" or "gamLoess" go up to 0.93 in accuracy, better than "ranger"= 0.90, but they have a greater variation in results when computing with 2 different datasets. So, I prefer to choose the minimum variation between different datasets, the most stable model.

In a next job, we need to go down the predicted error and we could also consider more advanced Machine Learning algorithms (specific classification models) that surely will improve the results with more tuning parametrs to fit with our dataset and also we could try others bigger datasets.

My next step in artificial intelligence will be the study of digitized images of magnetic resonance, tomography, radiography and other types of imaging studies to build an algorithm capable of diagnosing, without human intervention, the existence of a disease in the patient, with precision. equal to or greater than what the medical specialist can discover.

It would also be interesting to try other machine learning systems such as Keras, Tensor Flow, neural networks, etc.

References

https://rafalab.github.io/dsbook/

https://www.mayoclinic.org/diseases-conditions/heart-disease/symptoms-causes/syc-20353118

https://en.wikipedia.org/wiki/Cardiovascular_disease

https://www.sciencedaily.com/releases/2019/01/190131084238.htm

https://livebook.datascienceheroes.com

https://topepo.github.io/caret/index.html

https://cran.r-project.org/web/packages/ranger/ranger.pdf