# Report on Hearth Disease Project

Benedikt Bosch

8th August 2020

## Overview over the Heart Disease project

For this capstone project within Harvard's Professional Certificate in Data Science, I created a heart disease prediction model using the Heart Disease dataset and all the tools shown throughout the courses in this certificate. Within this project, I trained a machine learning algorithm using a training subset of the Heart Disease dataset in order to predict the prevalence of heart disease in the complementary testing set. Doing so, my model was able to yield an **accuracy** of **0.92** by using a **logistic regression model**.

## Heart Disease dataset

The original Heart Disease dataset was generated by the Hungarian Institute of Cardiology, the University Hospital Zurich, the University Hospital Basel, the V.A. Medical Center Long Beach and the Cleveland Clinic Foundation. It includes the diagnosis data of heart disease (angiographic disease status) for 303 patients, splitted into 13 different variables. This dataset will be referred to in the following as Heart Disease datset. It can be found under the following link: https://archive.ics.uci.edu/ml/datasets/Heart+Disease

## Approach and analysis overview

I developed my algorithm using the test set (80% of the data of the Heart Disease dataset). To test the algorithm, I predicted the prevalence of heart disease in the test set (20% of the data of the Heart Disease dataset) as if it was unknown. To evaluate how close my predictions were to the true values in the test set, accuracy (% of correct predictions) was used as loss function.

## Data cleaning

In a first step, after downloading the raw data, I inspected the head of the disease raw data.

```
kable(heart_renamed[1:6, ])
```

| age | sex | cp | bp | chol | bs | electro | max_hr | ang | STd | sp | ves | def | dis |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 63 | 1 | 1 | 145 | 233 | 1 | 2 | 150 | 0 | 2.3 | 3 | 0 | 6 | 0 |
| 67 | 1 | 4 | 160 | 286 | 0 | 2 | 108 | 1 | 1.5 | 2 | 3 | 3 | 2 |
| 67 | 1 | 4 | 120 | 229 | 0 | 2 | 129 | 1 | 2.6 | 2 | 2 | 7 | 1 |
| 37 | 1 | 3 | 130 | 250 | 0 | 0 | 187 | 0 | 3.5 | 3 | 0 | 3 | 0 |
| 41 | 0 | 2 | 130 | 204 | 0 | 2 | 172 | 0 | 1.4 | 1 | 0 | 3 | 0 |
| 56 | 1 | 2 | 120 | 236 | 0 | 0 | 178 | 0 | 0.8 | 1 | 0 | 3 | 0 |

The excerpt of the raw data revealed that the following 13 variables were available to train the machine learning algorithm in predicting whether heart disease was present or absent.

1. Sex
2. Age
3. Chest pain type
4. Resting blood pressure
5. Serum cholestoral in mg/dl
6. Fasting blood sugar > 120 mg/dl
7. Resting electrocardiographic results
8. Maximum heart rate
9. Exercise induced angina
10. ST depression induced by exercise relative to rest
11. Slope of the peak exercise ST segment
12. Number of major vessels
13. Defect type

The last column "*disease*" indicates the angiographic disease status, differentiated by severeness from 0 to 4. For the purpose of this capstone project, I only attempted to distinguish between presence (values 1,2,3,4) and absence (value 0) of a heart disease.

```
heart <- heart %>%
  mutate(disease = ifelse(disease > 0, 1, 0))
```

In a next step, I tried to Identify whether the data is tidy or not. For this purpose, I had a look whether there any *nas* in the data.

In total, there were 6 rows of patient data with na values, so I omitted these rows.

```
heart <- heart %>%
  na.omit()
```

Before exploring the data in more detail, the data classes of the columns needed to be adjusted. When downloading the data, all colums were imported as numeric variables. However, a closer look revealed that the sex, chest_pain, blood_sugar, electrocardiography, exercise_angina, slope_peak, major_vessels, defect and disease columns are in fact factors. Consequently, those values are discrete variables whereas the other ones are continous ones.

After tidying the data (297 observations à 13 variables remaining), the Heart disease dataset looks as following:

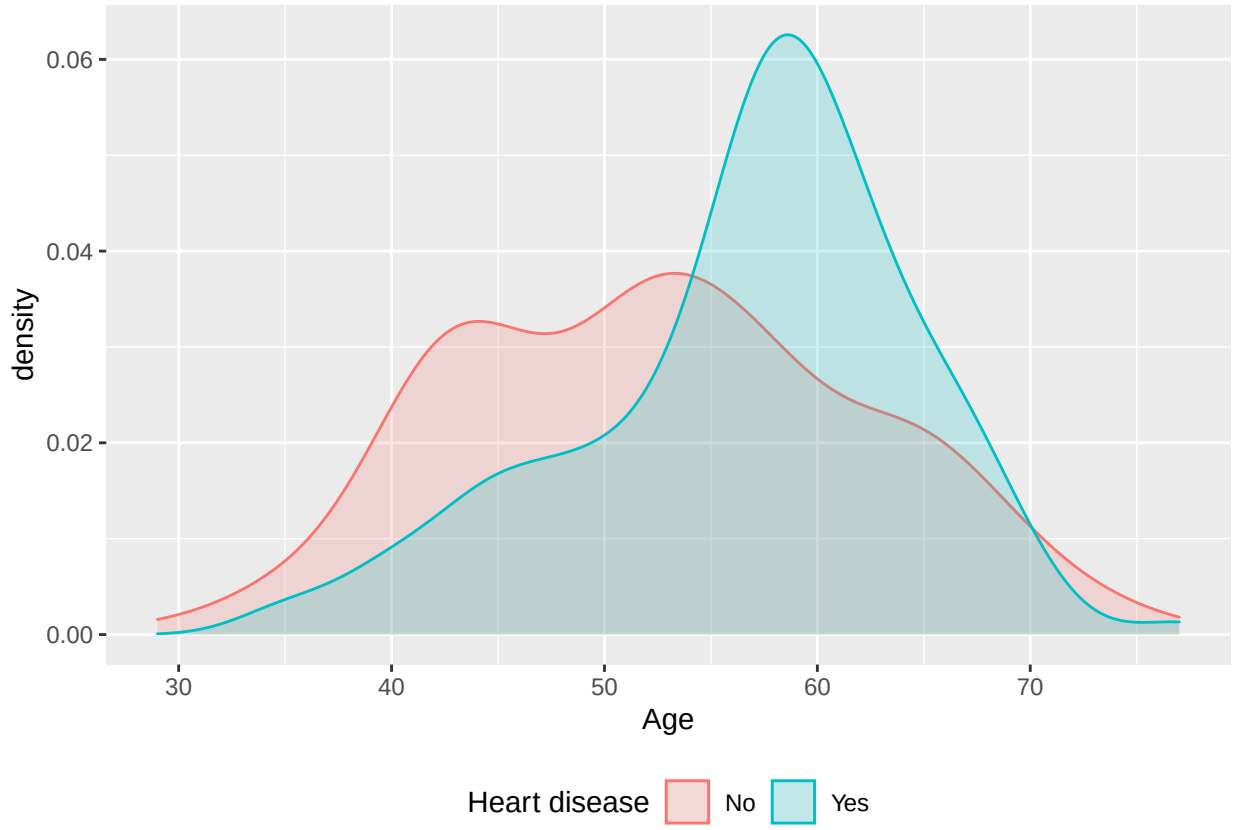
```
kable(heart_renamed[1:6, ])
```

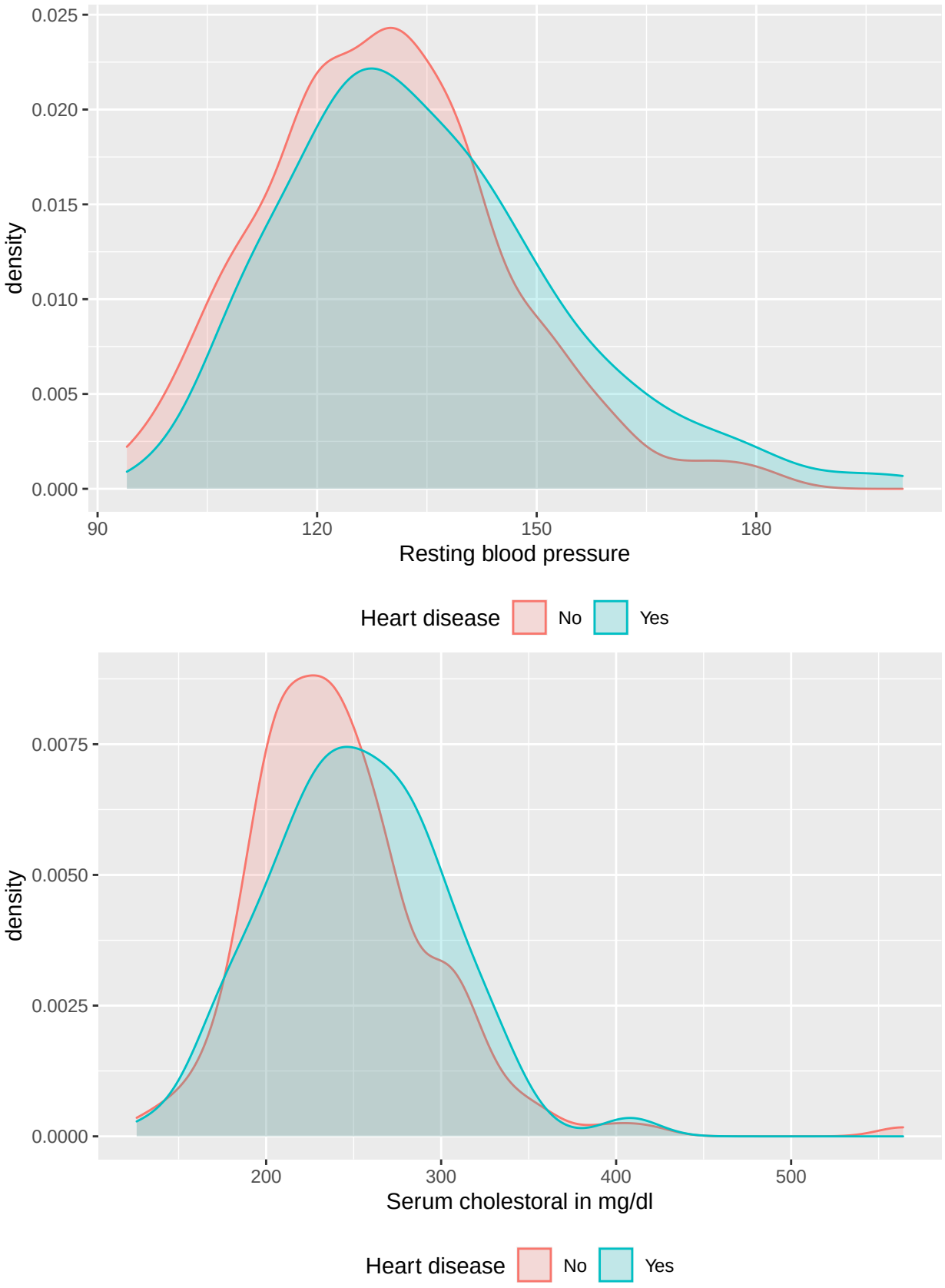
| age | sex | cp | bp | chol | bs | electro | max_hr | ang | STd | sp | ves | def | dis |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 63 | 1 | 1 | 145 | 233 | 1 | 2 | 150 | 0 | 2.3 | 3 | 0 | 6 | 0 |
| 67 | 1 | 4 | 160 | 286 | 0 | 2 | 108 | 1 | 1.5 | 2 | 3 | 3 | 1 |
| 67 | 1 | 4 | 120 | 229 | 0 | 2 | 129 | 1 | 2.6 | 2 | 2 | 7 | 1 |
| 37 | 1 | 3 | 130 | 250 | 0 | 0 | 187 | 0 | 3.5 | 3 | 0 | 3 | 0 |
| 41 | 0 | 2 | 130 | 204 | 0 | 2 | 172 | 0 | 1.4 | 1 | 0 | 3 | 0 |
| 56 | 1 | 2 | 120 | 236 | 0 | 0 | 178 | 0 | 0.8 | 1 | 0 | 3 | 0 |

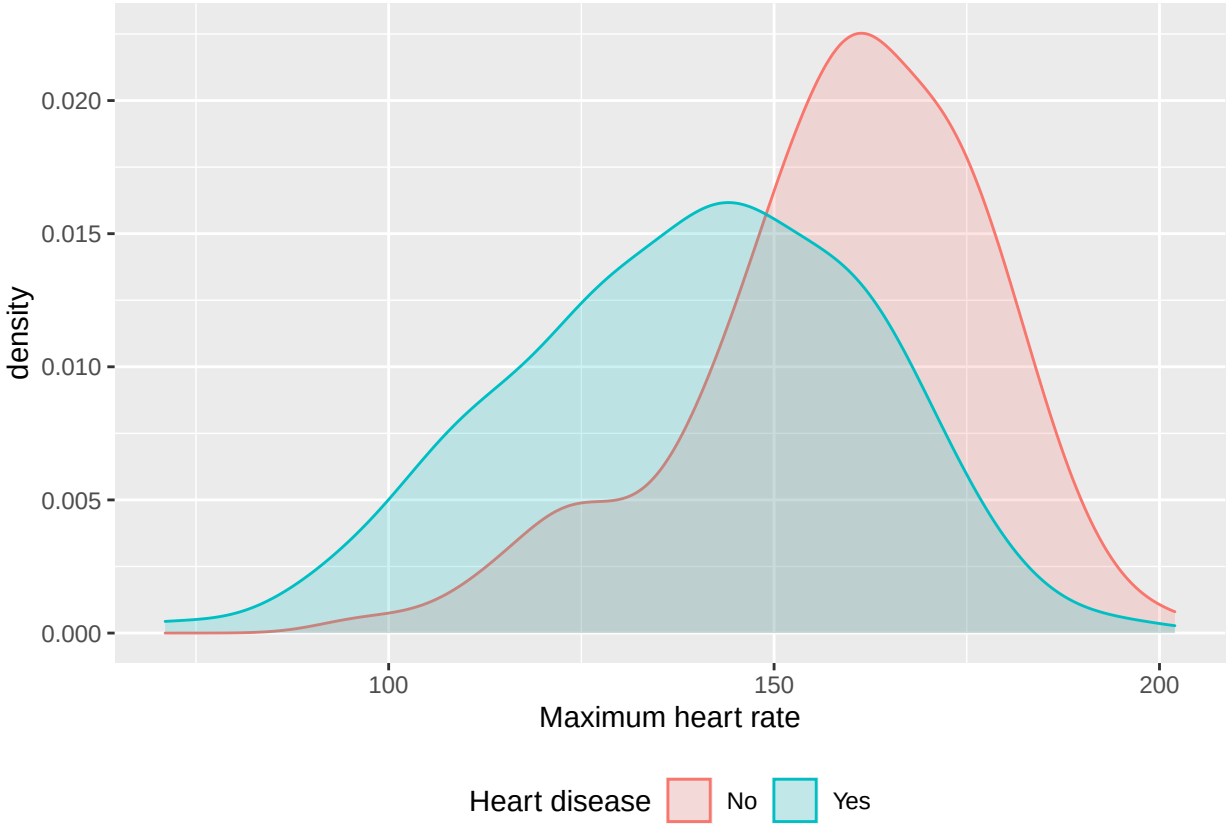
## Insights gained through data exploration and visualization

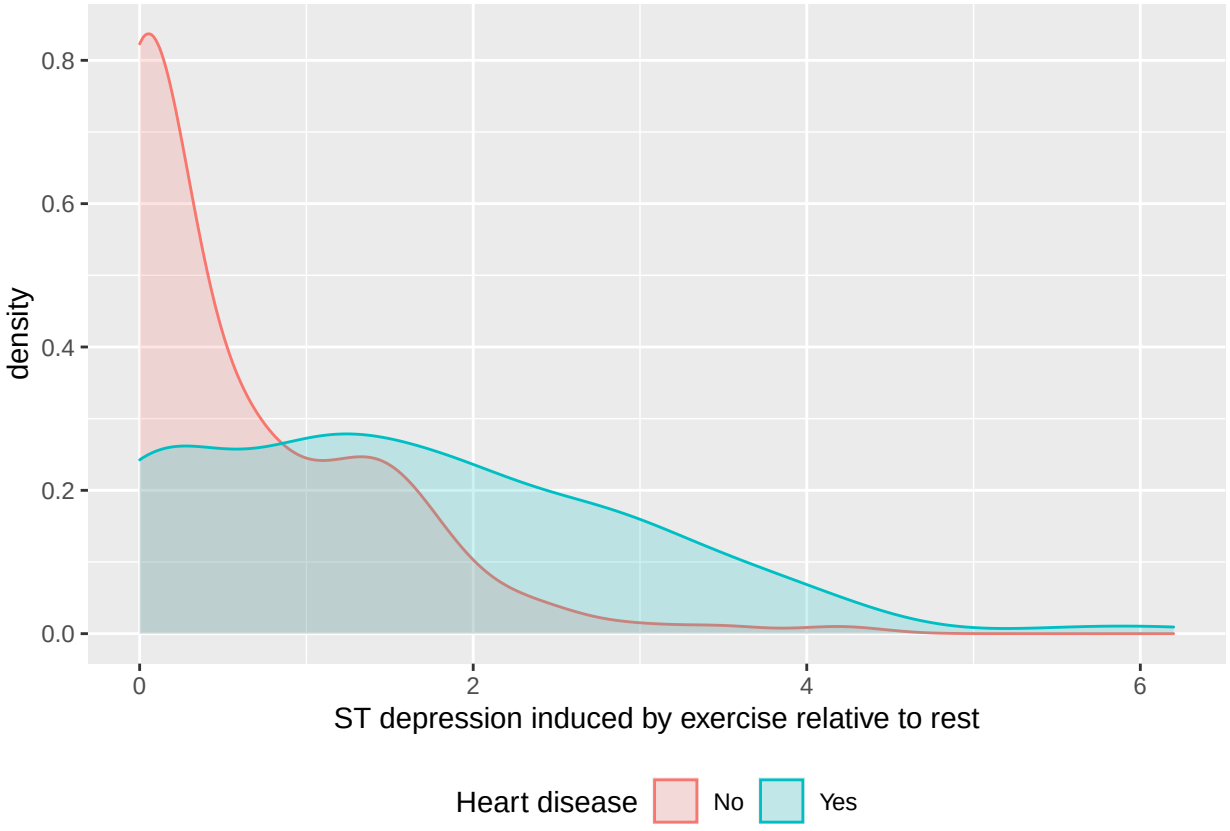
For the data exploration part of this project, I had a look at the distribution of each variable in regard to the patient's disease presence in the Heart Disease dataset. First, I had a look at the continous variables by displaying their density distribution.

### Continous variables' distribution




0.025
0.020
0.015
0.010
0.005
0.000
density
90
120
150
180
Resting blood pressure
Heart disease
No
Yes
0.0075
0.0050
0.0025
0.0000
density
200
300
400
500
Serum cholestoral in mg/dl
Heart disease
No
Yes

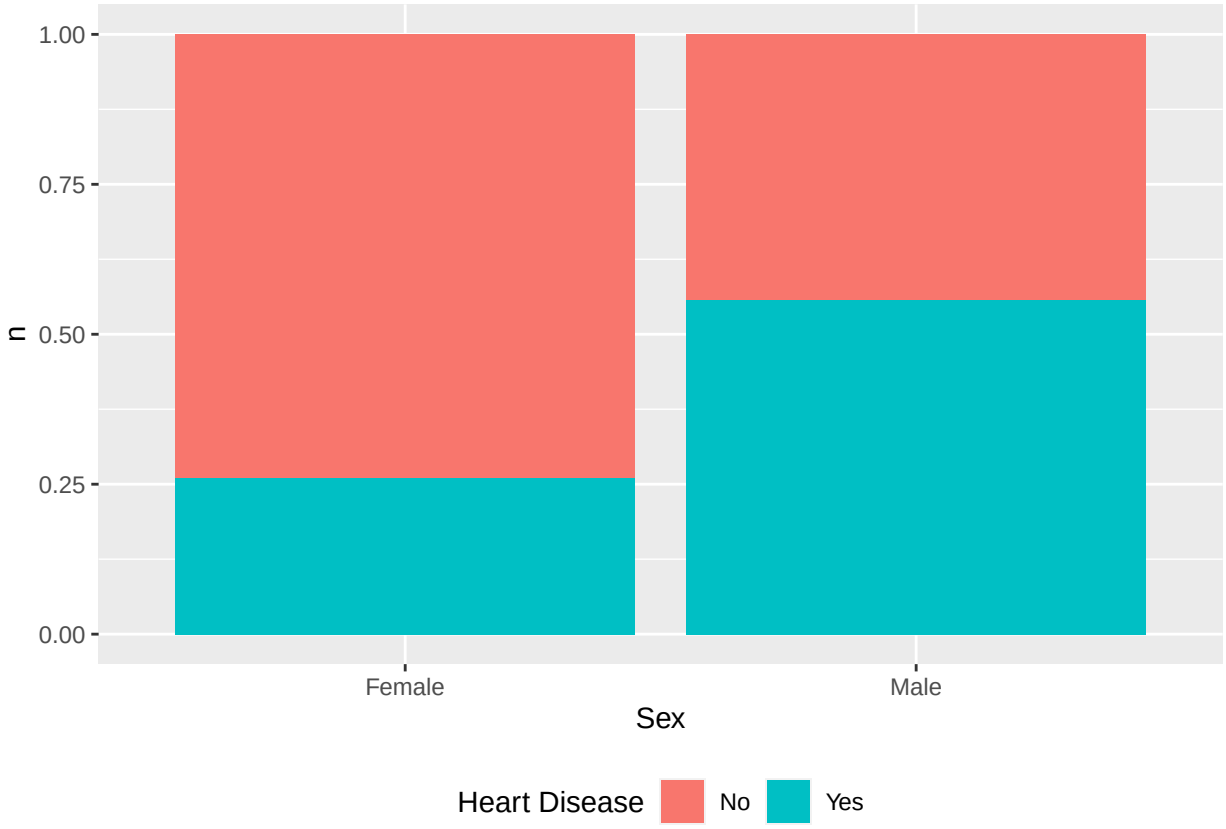

density
0.020
0.015
0.010
0.005
0.000
100
150
200
Maximum heart rate
Heart disease
No
Yes

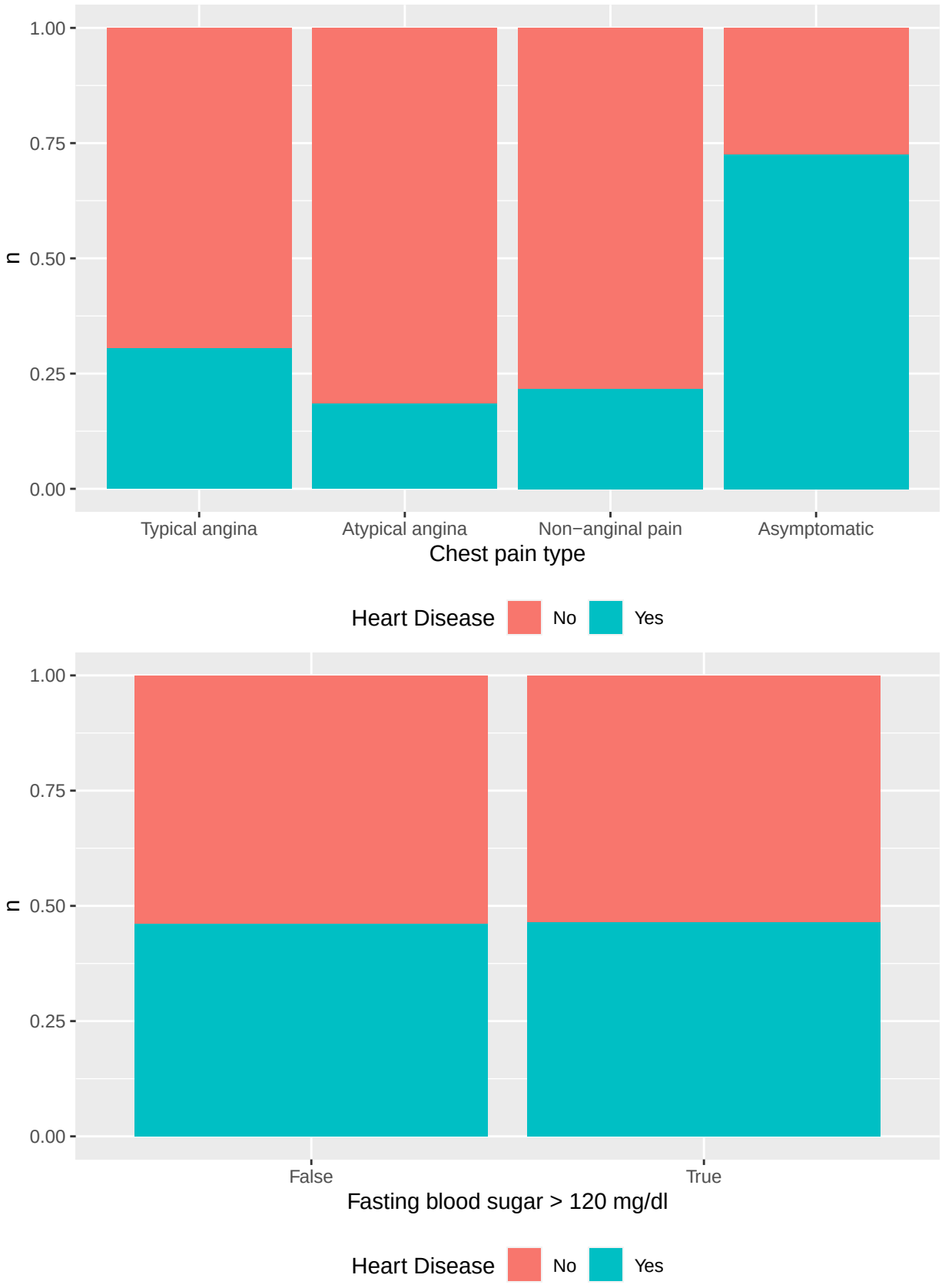

density
0.8
0.6
0.4
0.2
0.0
0
2
4
6
ST depression induced by exercise relative to rest
Heart disease
No
Yes

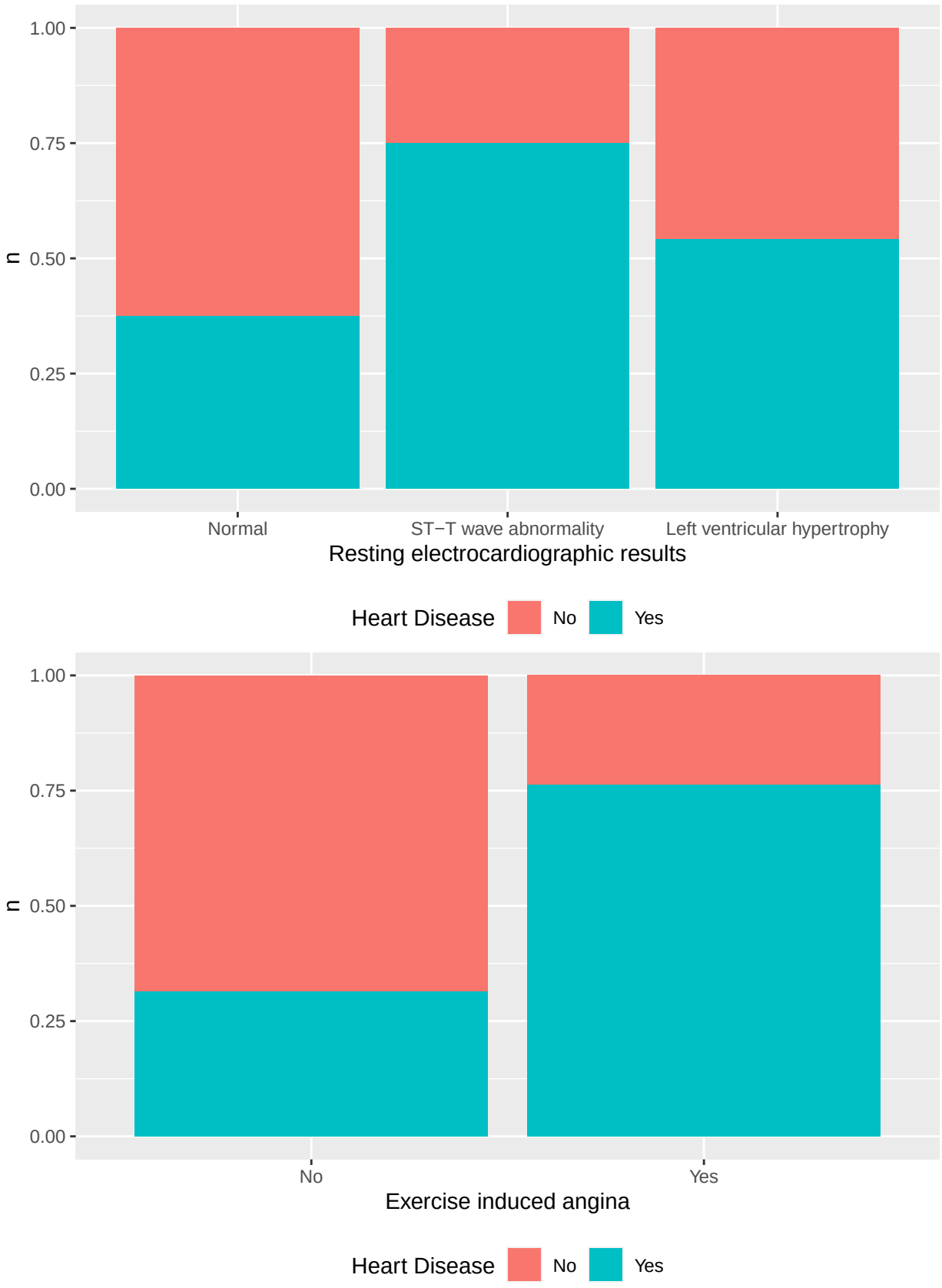
Looking at the five graphs, it was discernible that only the maximum heart rate has significant predictive power to differentiate between presence and absence of a heart disease.

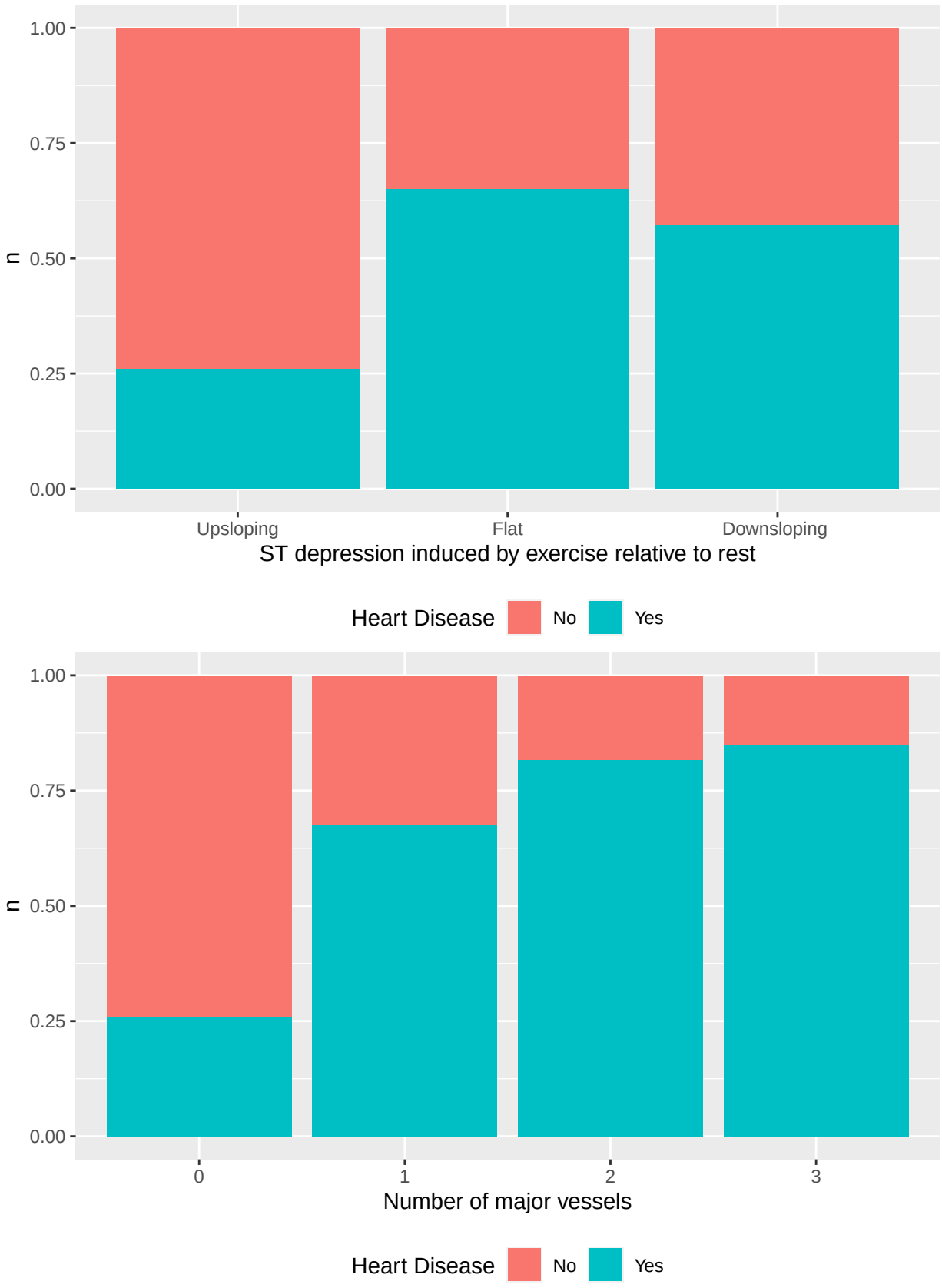
### Discrete variables' distribution

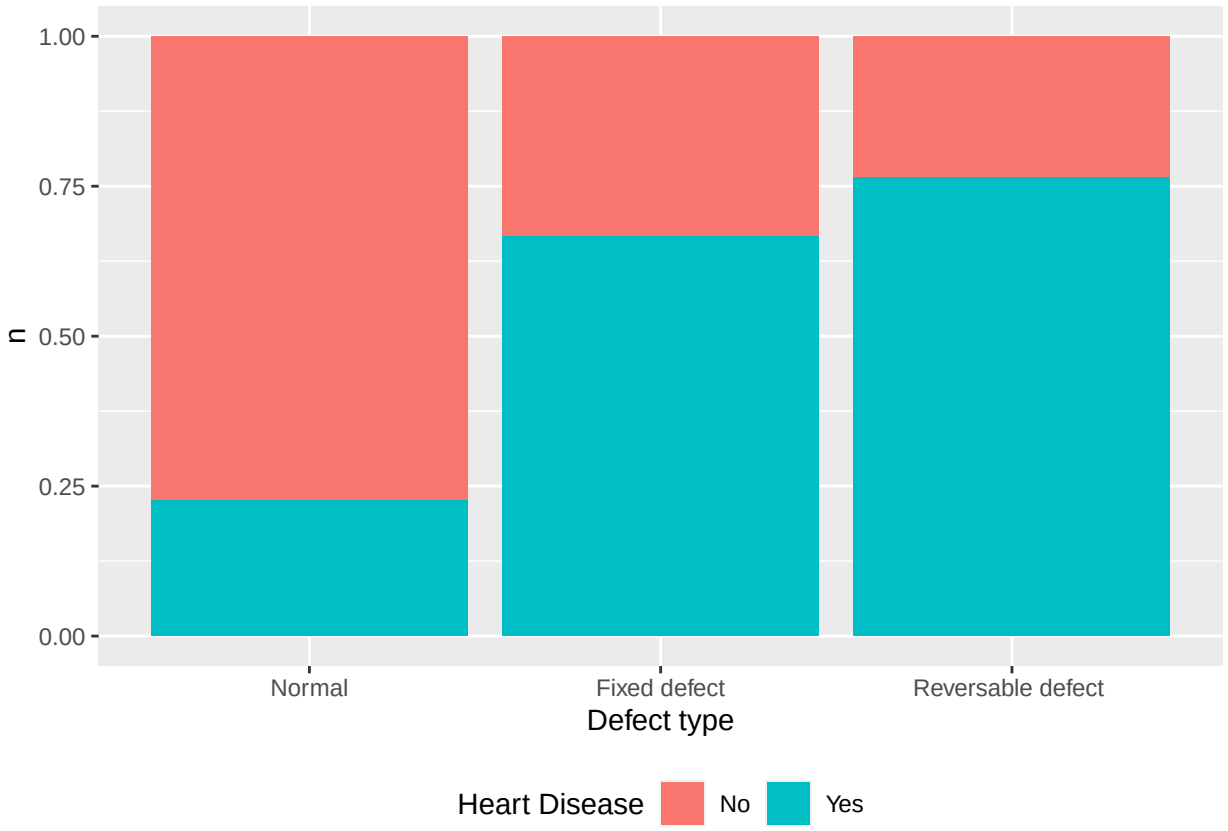
Second, I had a look at the discrete variables by plotting their distribution in a bar chart.




1.00
0.75
0.50
0.25
0.00
n
Typical angina
Atypical angina
Non−anginal pain
Asymptomatic
Chest pain type
Heart Disease
No
Yes
1.00
0.75
0.50
0.25
0.00
n
False
True
Fasting blood sugar > 120 mg/dl
Heart Disease
No
Yes


1.00
0.75
0.50
0.25
0.00
n
Normal
ST−T wave abnormality
Left ventricular hypertrophy
Resting electrocardiographic results
Heart Disease
No
Yes
1.00
0.75
0.50
0.25
0.00
n
No
Yes
Exercise induced angina
Heart Disease
No
Yes


1.00
0.75
0.50
0.25
0.00
n
Upsloping
Flat
Downsloping
ST depression induced by exercise relative to rest
Heart Disease
No
Yes
1.00
0.75
0.50
0.25
0.00
n
0
1
2
3
Number of major vessels
Heart Disease
No
Yes




Looking at the eight graphs, it was discernible that the sex, the chest pain type, the exercise induced angina, the slope of the peak exercise ST segment, the number of major vessels and the defect type - but not the fasting blood sugar level or the resting electrocardiographic results - have significant predictive power to differentiate between presence and absence of a heart disease.

As a consequence of the data exploration, I decided to only keep variable columns that have significant predictive power to differentiate between presence and absence of a heart disease. These are the following seven variables:

1. Sex
2. Chest pain type
3. Maximum heart rate
4. Exercise induced angina
5. Slope of the peak exercise ST segment
6. Number of major vessels
7. Defect type

For all seven factors, biases could be clearly demonstrated: men are more likely to suffer from heart disease, asymptomatic chest pain is a likely indicator of a heart disease, persons with a maximum heart rate of more than 180 beats per minute rarely need to fear a heart disease, exercise induced angina significantly increases the risk of a heart disease, an upsloping peak exercise ST segment, zero major vessels and a normal defect type reduce disease risk. Consequently, an appropriate machine learning algorithm needs to take into account those factors.

## Creation of train and validation sets

Before training the machine learning algorithm, a train and test set needed to be created from the Heart Disease data. The test set contained 20% of the Heart Disease data and was only used to to assess how close my predictions with different models were to the true values. The train set comprised 80% of the data and was used to train the different algorithms.

After cleaning the data and creating the train and test set, the two datsets comprised the follwing number of ratings:

| Dataset | Ratings |
|---|---|
| Train set | 237 |
| Test set | 60 |

## Modeling approach

A simple linear regression is normally a useful baseline approach but is often insufficiently flexible for more complex analyses. This is the case here. Therefore, to model the heart disease prediction system, I decided to use four different machine learning algorithms.

1. Logistic regression model
2. K-nearest neighbor model
3. Regression tree model
4. Random forest model

All four models are discriminative approaches that estimate the conditional probability directly and do not consider the distribution of the predictors. Generative models, to the contrary, can be very powerful when we the joint distribution of predictors conditioned on each class can successfully be estimated, which is not the case here. LDA (linear discriminant analysis) and QDA (quadratic discriminant analysis), for example, are not meant to be used with datasets that have many predictors as the number of parameters that need to be estimates becomes too large. Once the number of parameters approaches the size of the original data, the methods become impractical due to overfitting.

Each model was trained using the train set. After training the model, the presence of heart disease per patinet in the test set was predicted and the resulting accurcy calculated. Comparing the resulting accuracies, it was apparent that the logistic regression model is the best machine learning algorithm for this purpose.

### Logistic regression model

The baseline of the algorithm is the logistic regression model. It is the simplest possible recommendation system for the purpose of predicting categorical data: presence or absence of heart disease. Logistic regression is an extension of linear regression that assures that the estimate of conditional probability is between 0 and 1. This approach makes use of the logistic transformation by transforming probabilities to log odds. The odds tell you how much more likely something will happen compared to not happen.

$$g(p) = log\frac{p}{1-p}$$

```r
# compute fit of logistic regression algorithm
set.seed(20, sample.kind="Rounding")
glm_fit <- train(disease ~ ., data = train_set, method = "glm", family = "binomial")

# calculate predicted presence of heart disease (with glm)
glm_predicted_num <- predict(glm_fit, test_set)

# use logistic regression prediction to compute accuracy on the test set
glm_accuracy <- accuracy(glm_predicted_num, test_set$disease)
```

```
## Warning: `data_frame()` is deprecated as of tibble 1.1.0.
## Please use `tibble()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

Logistic regression forces our predictions to be on a plane and our boundary to be a line. This implies that a logistic regression model has no chance of capturing a non-linear nature. For this reason, logistic regression is often limited and not flexible enough to be useful. However, the logistic regression model yields an accuracy of 0.91667, the highest one of the four models.

### K-nearest neighbor model

K-nearest neighbors (kNN) estimates the conditional probabilities in a similar way to bin smoothing. However, kNN is easier to adapt to multiple dimensions. The general idea of smoothing is to group data points into strata in which the predicted value, here the disease status, can be assumed to be constant. This assumption can be made because the predicted value changes slowly and, as a result, is almost constant in small windows of time.

This assumption implies that a good estimate for the predicted value, $\hat{y}$ is the average of the predicted values, $Y_i$ in the window. This estimate can be expressed mathematically as follows:

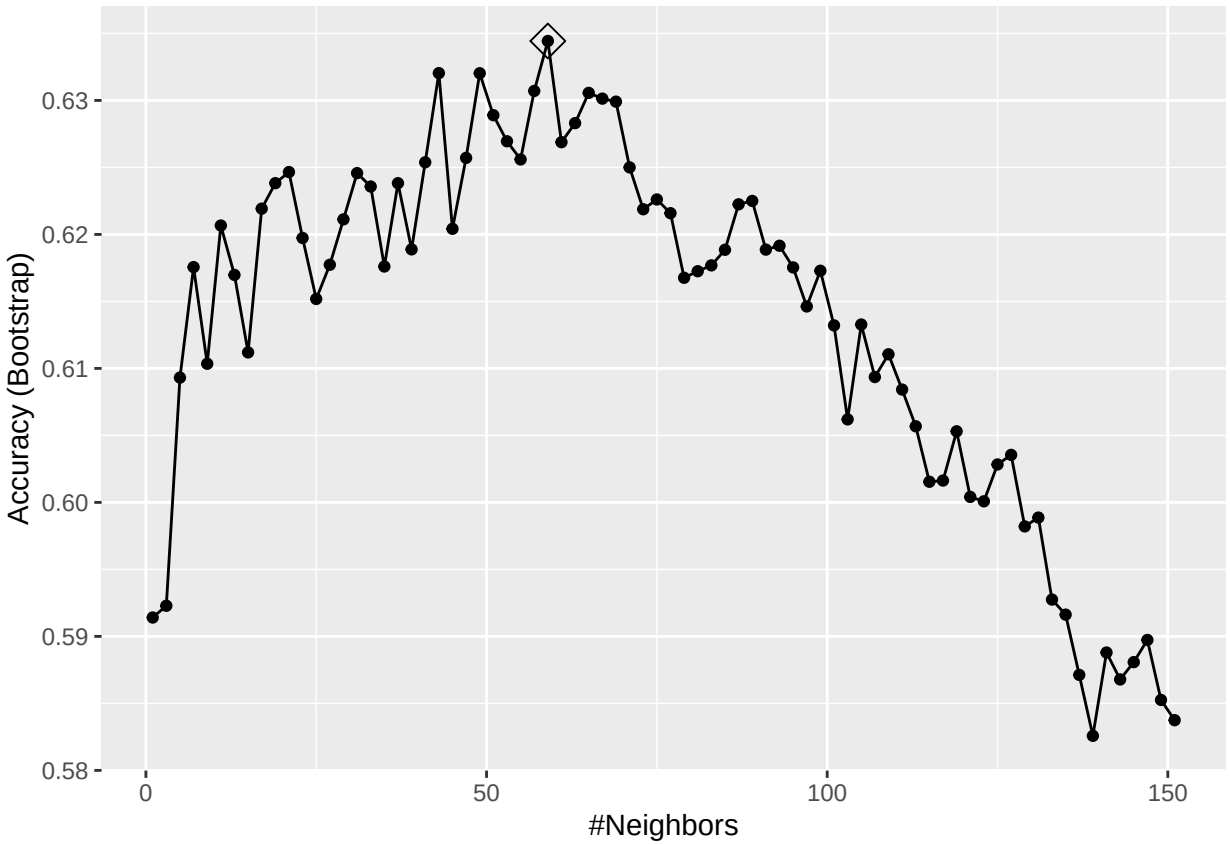
$$\hat{y} = \frac{1}{N_0} \sum_{i \in A_0} Y_i$$

In smoothing, we call the size of the interval satisfying the particular condition the neighborhood.

```r
# compute fit of knn algorithm
set.seed(20, sample.kind="Rounding")
knn_fit <- train(disease ~ ., data = train_set, method = "knn",
                 tuneGrid = data.frame(k = seq(1, 151, 2)))

# calculate predicted presence of heart disease (with knn)
knn_predicted_num <- predict(knn_fit, test_set)

# use knn prediction to compute accuracy on the test set
knn_accuracy <- accuracy(knn_predicted_num, test_set$disease)
```




From the graph it is discernible that the value of k that optimizes the model is 59, so that the interval to the left and to the right includes the 29 nearest neighbors on each side.

With kernel methods such as k-nearest neighbor, the optimal neighborhood to include a given percentage of the data becomes too large when multiple predictors are used, such as in this analysis. With larger neighborhoods, the method loses flexibility. That is why the k-nearest-neigbor model performs significantly worse than the logistic regression model, it only achives an accuracy of 0.68333, the lowest one of all four models.

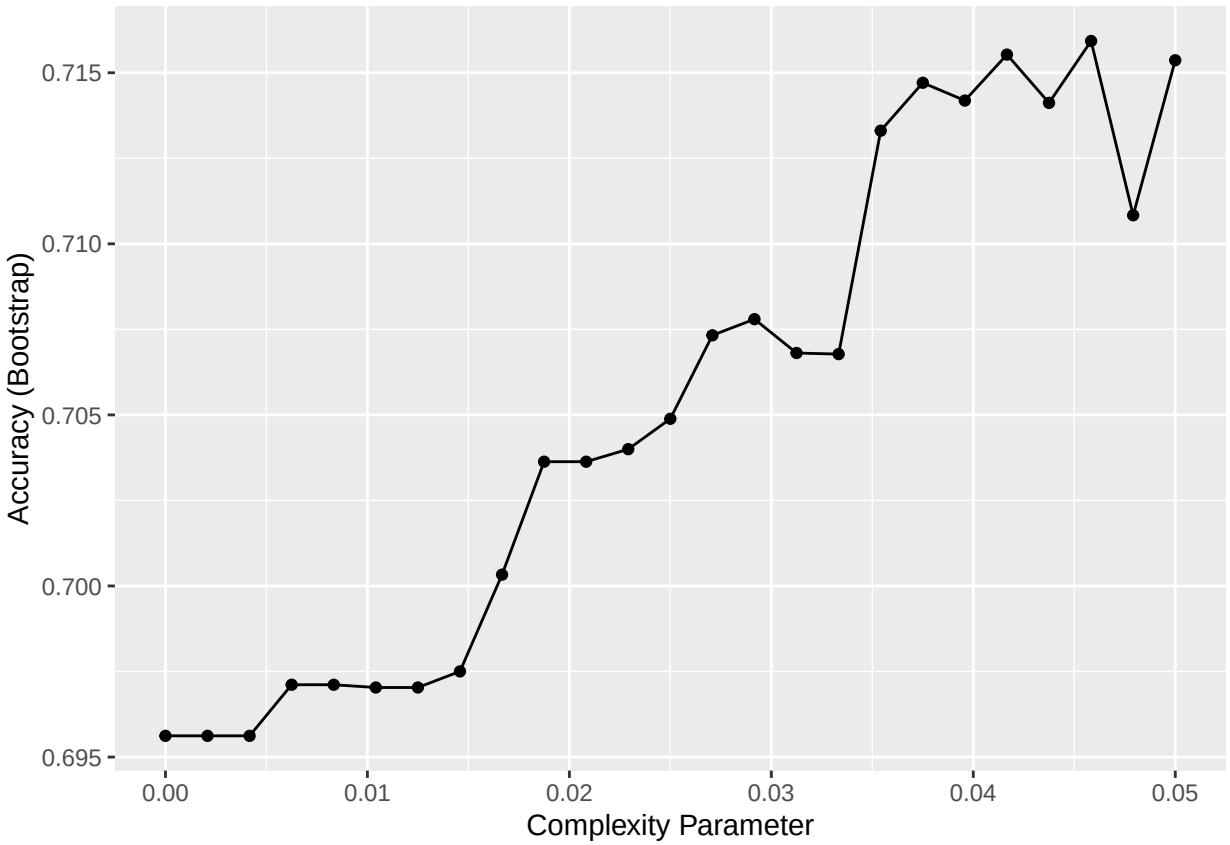
## Regression tree model

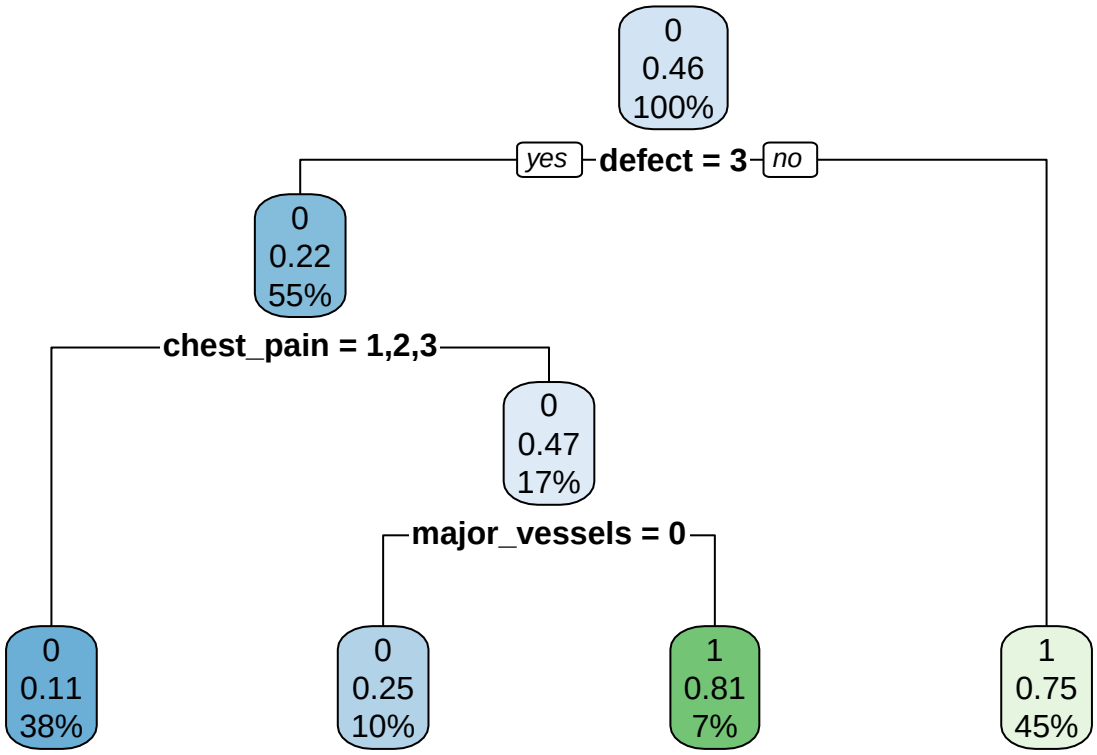
Regression trees operate by predicting an outcome variable by partitioning the predictor space. However, this partioning bears the risk of overtraining. To avoid this situation, my algorithm sets a minimum for how much the residual sum of squares must improve for another partition to be added. This parameter is referred to as the complexity Parameter.

```
# compute fit of regression tree algorithm
set.seed(20, sample.kind="Rounding")
rpart_fit <- train(disease ~ ., data = train_set, method = "rpart",
                   tuneGrid = data.frame(cp = seq(0, 0.05, len = 25)))

# calculate predicted presence of heart disease (with rpart)
rpart_predicted_num <- predict(rpart_fit, test_set)

# use regression tree prediction to compute accuracy on the test set
rpart_accuracy <- accuracy(rpart_predicted_num, test_set$disease)
```


Accuracy (Bootstrap)
0.715
0.710
0.705
0.700
0.695
0.00
0.01
0.02
0.03
0.04
0.05
Complexity Parameter


0
0.46
100%
yes
defect = 3
no
0
0.22
55%
chest_pain = 1,2,3
0
0.47
17%
major_vessels = 0
0
0.11
38%
0
0.25
10%
1
0.81
7%
1
0.75
45%

The main advantage of regression trees is that they are highly interpretable and easy to visualize. They can model human decision processes and do not require use of dummy predictors for categorical variables.

From the graph we can see that the value of the complexity paramter that optimizes the model is 0.0458333333333333. The decision tree illustrates how the algorithm predicts the presence or absence of a heart disease based on available variables. In terms of accuracy, regression trees are rarely the best performing method. The regression tree yields an accuracy of 0.73333 which is an improvement compared to the k-nearest neighbor model but still significantly worse than the logistic regression.

Regression trees may not be the best performing method since they are not very flexible, and they are quite susceptible to changes in the training data.

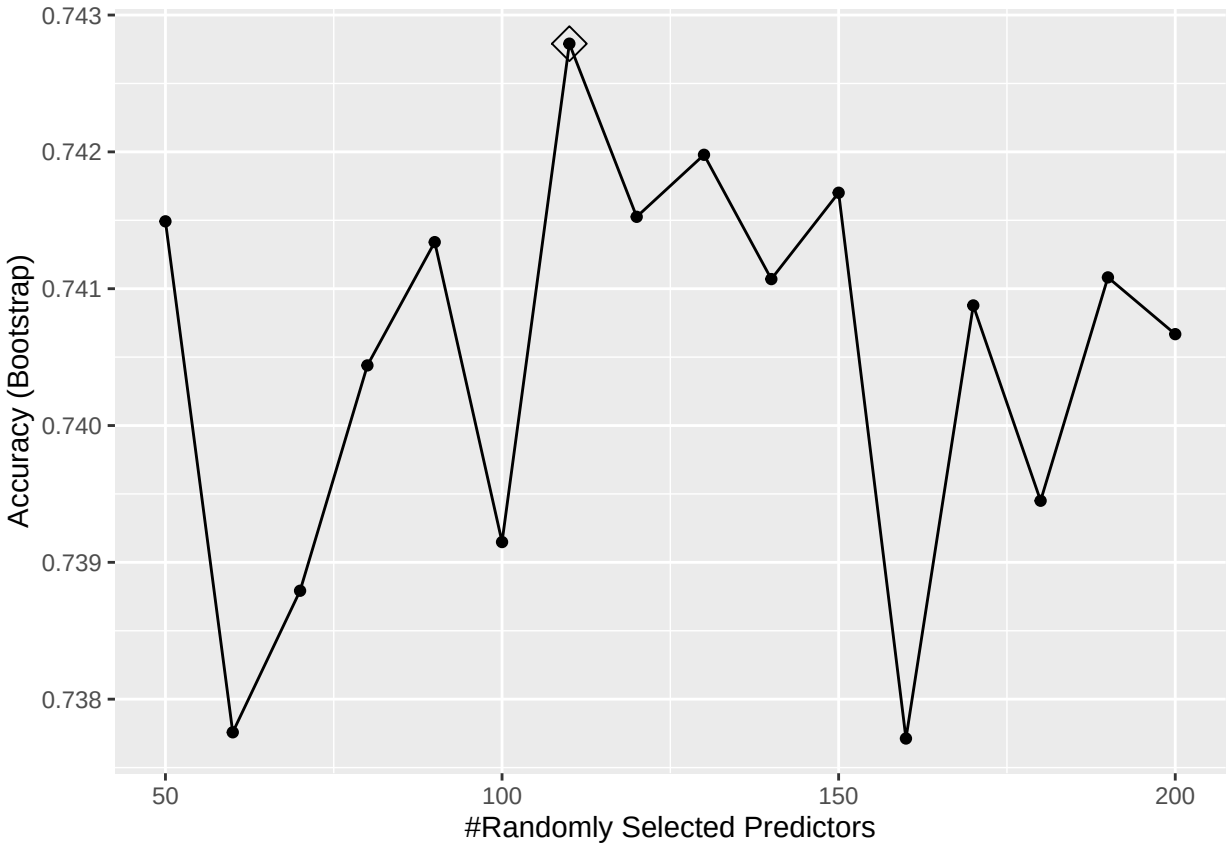
## Random forest model

In a last step, I ran a random forest model. Random forests are a very popular machine learning approach that addresses the shortcomings of regression trees. The goal is to improve prediction performance and reduce instability by averaging multiple regression trees (a forest of trees constructed with randomness). Thereby, *mtry*, the number of variables randomly sampled as candidates at each split, is optimized.

```
# compute fit of random forest algorithm
set.seed(20, sample.kind="Rounding")
rf_fit <- train(disease ~ ., data = train_set, method = "rf",
                tuneGrid = data.frame(mtry = seq(50, 200, 10)))

# calculate predicted presence of heart disease (with rf)
rf_predicted_num <- predict(rf_fit, test_set)

# use random forest prediction to compute accuracy on the test set
rf_accuracy <- accuracy(rf_predicted_num, test_set$disease)
```




From the graph we can see that the value of *mtry* that optimizes the model is 110, thus a random sample of 110 at each split optimizes this model. Thereby, it yields an accuracy of 0.9, a clear improvement to the regression tree model and the second highest of the four models. One small disadvantage of random forests is that interpretability is lost.

## Conclusion

| method | accuracy |
|---|---|
| Logistic Regression | 0.91667 |
| K-Nearest Neighbor | 0.68333 |
| Regression Tree | 0.73333 |
| Random Forest | 0.90000 |

By testing four different models on the Health Disease dataset, I was able to achieve an accuracy of 0.91667 with a logistic regression model. The model outperformed the k-nearest neighbor, the regression tree and the random forest model, even though it is the simplest one. This brings me back to one of the key learnings of the course: If you can't beat it with a more complex approach, you probably simply want to stick to regression.

In the future, I aim to explore further machine learning algorithms that outperform the accuracy of the logistic regression model. I am more than happy about my achievements in the Professional Certificate in Data Science course and would strongly recommend it to anyone interested in data science and R.