Cardiovascular disease prediction report - HarvardX PH125.9x - Capstone

10-09-2021

Introduction

In my project I'm going to create machine learning models, which are capable of predicting presence of absence of cardiovascular disease.

Dataset

Our task is to create binary classification for presence (1) or absence (0) of cardiovascular disease. https://www.kaggle.com/sulianova/cardiovascular-disease-dataset

Our dataset is separated by ";", so we need to load it with:

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

Full dataset contains 70000 rows and 13 columns. Value of each column was described in table 1.

Table 1. Description of columns

Variable	Class	First_values	Description
id	integer	0, 1, 2, 3, 4, 8	Unique identification number
age	integer	18393, 20228, 18857, 17623, 17474, 21914	Age (days)
gender	integer	2, 1, 1, 2, 1, 1	Gender (1 - woman, 2 -man)
height	integer	168, 156, 165, 169, 156, 151	Height (cm)
weight	double	62, 85, 64, 82, 56, 67	Weight (kg)
ap_hi	integer	110, 140, 130, 150, 100, 120	Systolic blood pressure
ap_lo	integer	80, 90, 70, 100, 60, 80	Diastolic blood pressure
cholesterol	integer	1, 3, 3, 1, 1, 2	Cholesterol (1 - normal, 2 - above normal, 3 - well above normal)
gluc	integer	1, 1, 1, 1, 1, 2	Glucose (1 - normal, 2 - above normal, 3 - well above normal)
smoke	integer	0, 0, 0, 0, 0, 0	Smoking (0 - no, 1 - yes)
alco	integer	0, 0, 0, 0, 0, 0	Alcohol intake (0 - no, 1 - yes)
active	integer	1, 1, 0, 1, 0, 0	Physical activity (0 - no, 1 - yes)
cardio	integer	0, 1, 1, 1, 0, 0	Presence or absence of cardiovascular disease (0 - absence, 1 - presence)

1. Analysis

1.1. Preprocessing

All of the data is in good shape, but let's change age in days to age in years and remove column id, because it contains just incrementing numbers for each row.

```
# Change age from days to years
ds <- ds %>% mutate(age = round(age/365, digits = 0)) %>% select(-id)
```

Also data contains values of systolic blood pressure and diastolic blood pressure with different formating. Like 150 and 15000 - both means the same, so unify it, by following code:

```
ds <- ds %>%
  mutate(ap_hi = if_else(nchar(ap_hi) > 3,as.integer(ap_hi/100),ap_hi),
         ap_lo = if_else(nchar(ap_lo) > 3,as.integer(ap_lo/100),ap_lo))
```

After next examination there are still some values that are impossible, so let's preprocess it

```
ds <- ds %>%
  mutate(ap_hi = if_else(ap_hi > 250,as.integer(ap_hi/10),ap_hi),
         ap_lo = if_else(ap_lo > 250,as.integer(ap_lo/10),ap_lo))
```

So in our dataset there are still 0 values for blood preasure and values that are too low, remove them. Based on study Bambrick H. - Relationships between BMI, waist circumference, hypertension and fasting glucose: Rethinking risk factors in Indigenous diabetes (https://www.researchgate.net/publication/241225018_Relationships_between_BMI_waist_circumference_hypertension_and_fasting_glucose_Rethinking_risk_factors_in_Indigenous_diabetes) let's get data with ap_hi and ap_lo above 60.

```
ds <- ds[ds$ap_hi>60 & ds$ap_lo>60,]
```

Separate our dataset into learning set and validation set as 80/20.

Now our training contains 52806 rows and our validation set - 13201 rows.

First 6 rows of our preprocessed dataset are shown in table 1.1.

Table 1.1. Preprocessed dataset

	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio
1	50	2	168	62	110	80	1	1	0	0	1	0
2	55	1	156	85	140	90	3	1	0	0	1	1
3	52	1	165	64	130	70	3	1	0	0	0	1
4	48	2	169	82	150	100	1	1	0	0	1	1
6	60	1	151	67	120	80	2	2	0	0	0	0
7	61	1	157	93	130	80	3	1	0	0	1	0

1.2. EDA

Firstly check for missing values

	x
age	0
gender	0
height	0
weight	0
ap_hi	0
ap_lo	0
cholesterol	0
gluc	0
smoke	0
alco	0
active	0
cardio	0

Plot count of y value in your train set to see if it's imbalanced. Count of y for both class is shown in figure 1.1. As we can see is not imbalanced.

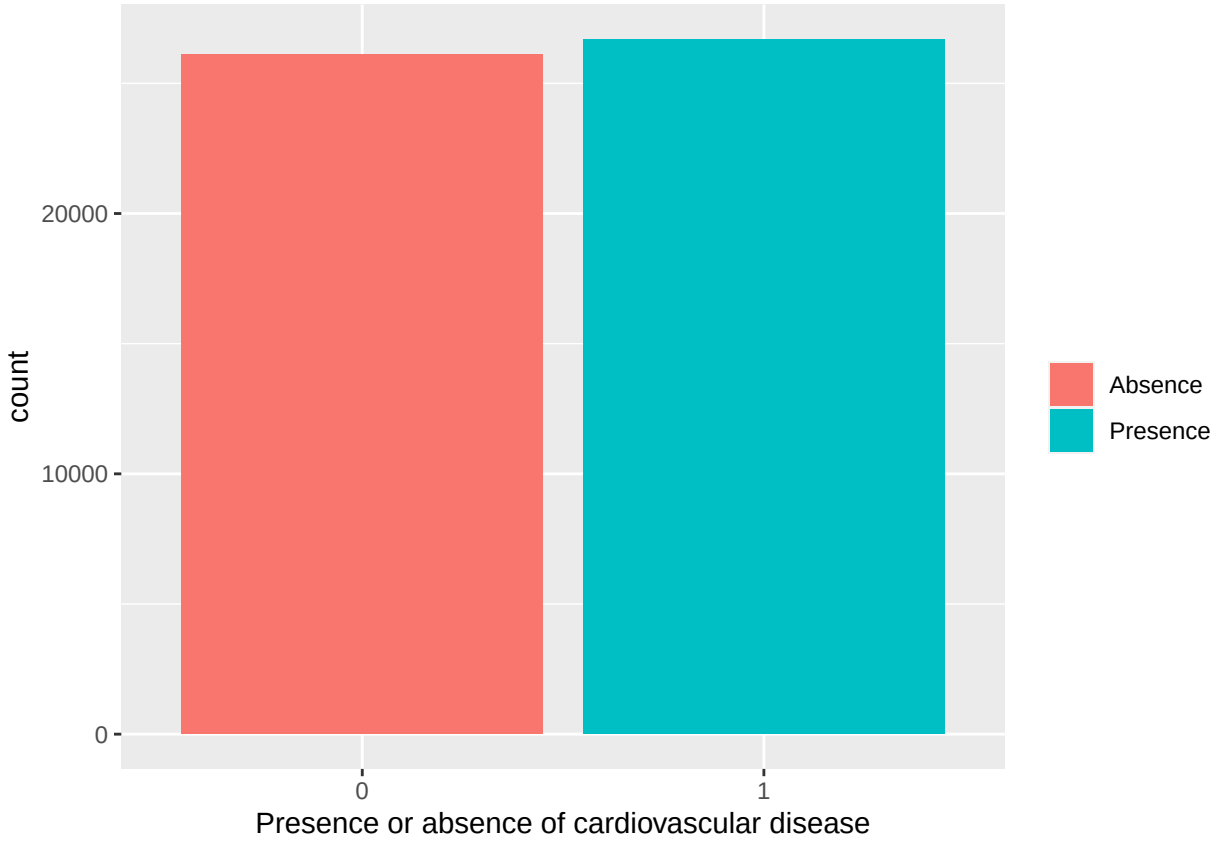

Fig. 1.1. Count of y for both class

Now let's check if in our dataset exists some correlation between presence of cardiovascular disease and age of patients. Density plot and boxplot of age and disease is shown in figure 1.2.

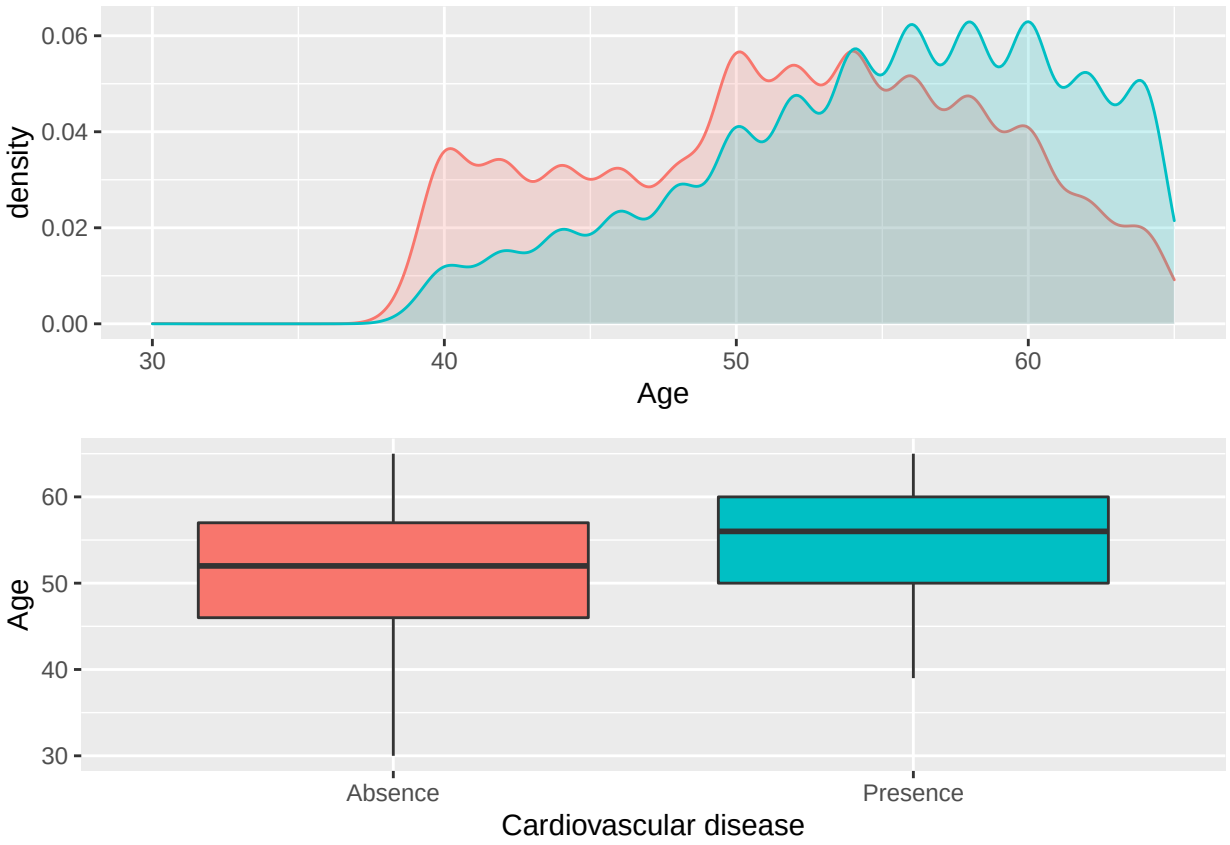

Fig. 1.2. Density plot and boxplot of age and disease

From the plot above we can see that older peoples are more likely to have a disease, but intervals are overlapping.

Now let's see a structure of gender based on cardiovascular disease, it is shown in figure 1.3

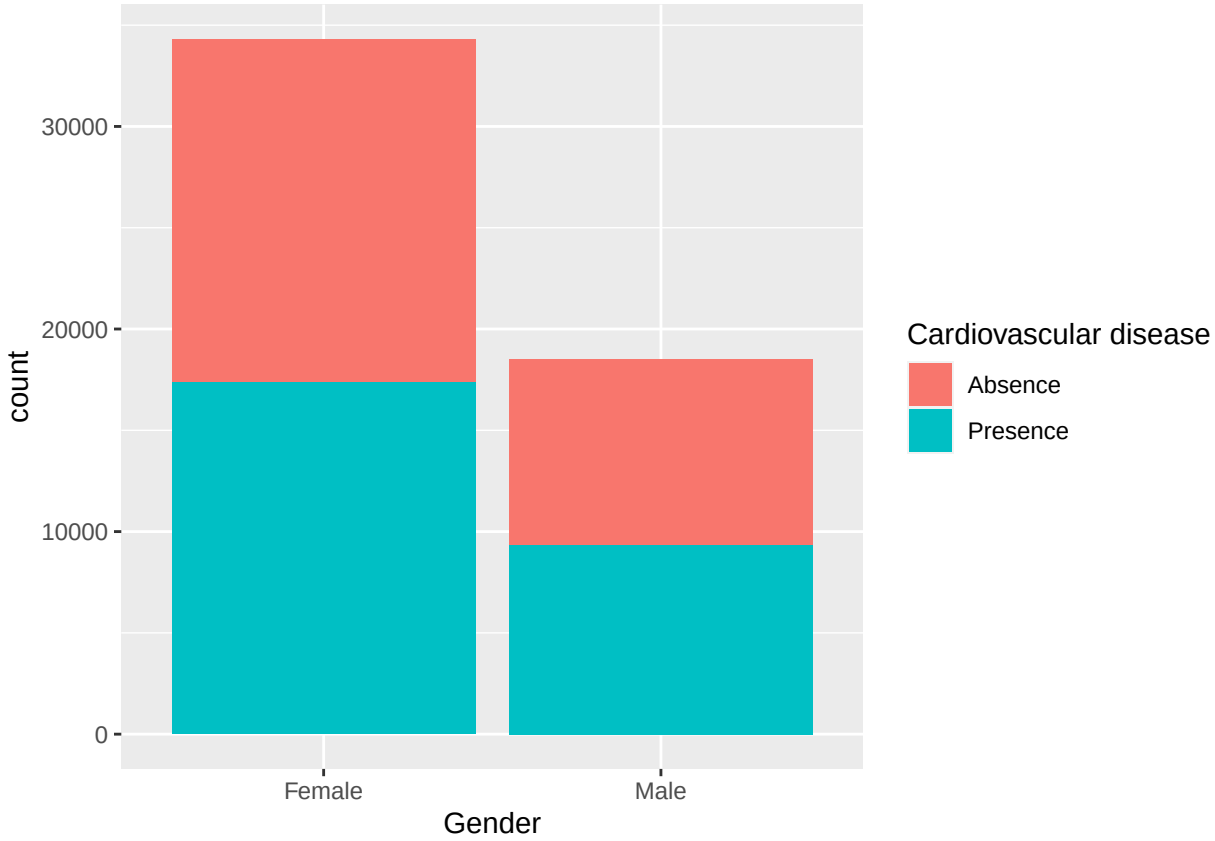

Fig. 1.3. Structure of gender based on cardiovascular disease

From the image above we can see that in our dataset is more females that males, but structure of cardiovascular disease between sexes in similar.

Weight and height can have influence of each other, so plot weight vs. height with color marking of cardiovascular disease status.

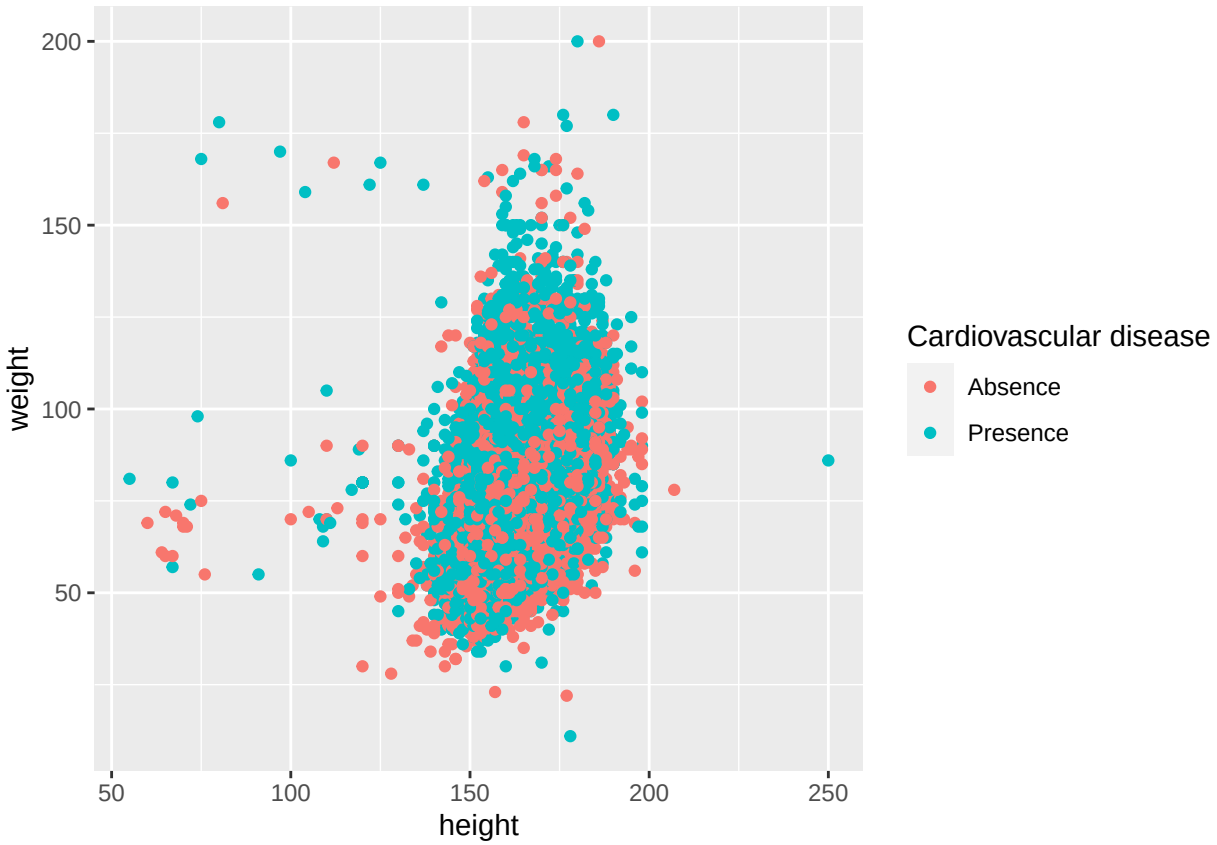

Fig. 1.4. Scatterplot height vs. weight with cardiovascular disease status

As we can see there is almost no correlation betwwen height, weight and cardiovascular disease status. But there is a dependence between weight, height and gender. It was show at figure 1.5.

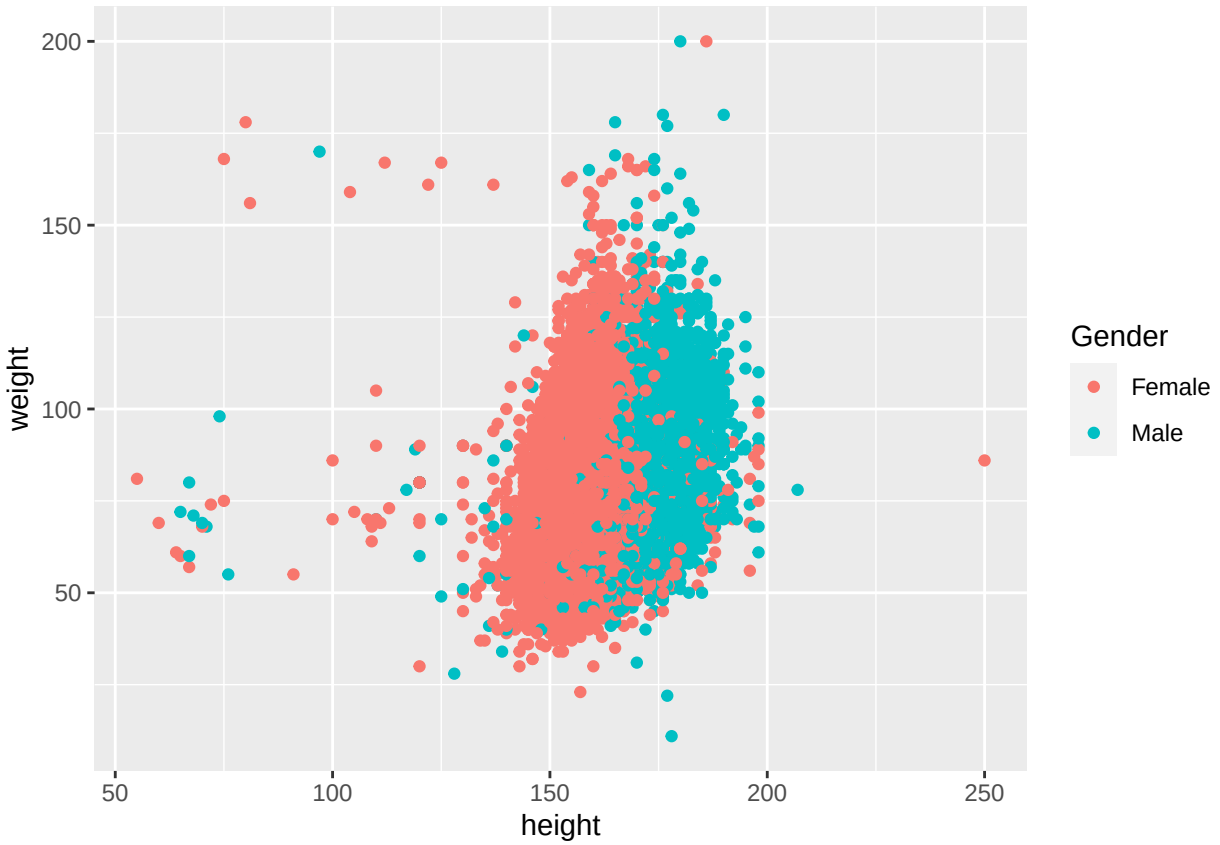

Figure. 1.5. Scatterplot height vs. weight with gender

Now let's plot a systolic blood pressure vs. diastolic blood pressure with cardiovascular disease status, it is shown in figure 1.6.

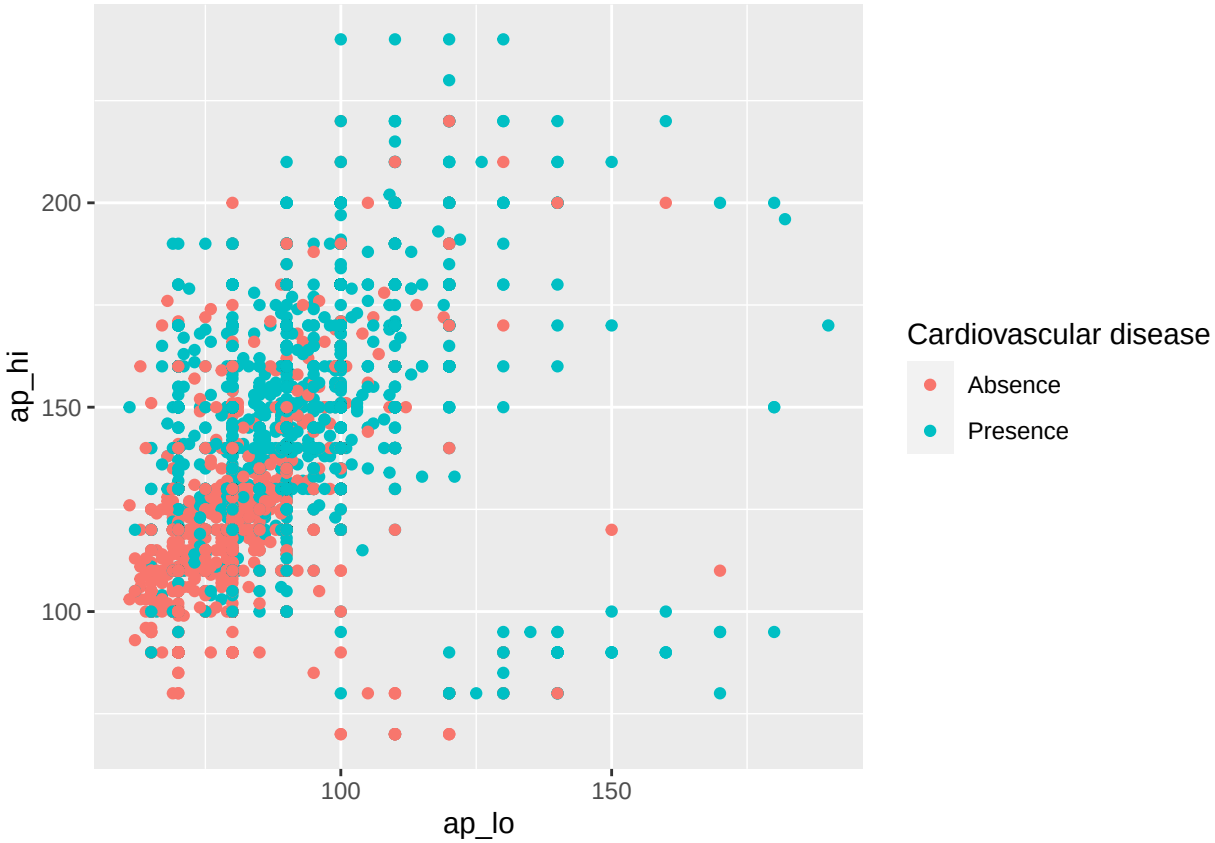

Figure. 1.6. Scatterplot systolic blood pressure vs. diastolic blood pressure with cardiovascular disease status

Additionaly let's examine boxplot for the same - figure 1.7. For most cases higher value of blood preasure means presence of disease.

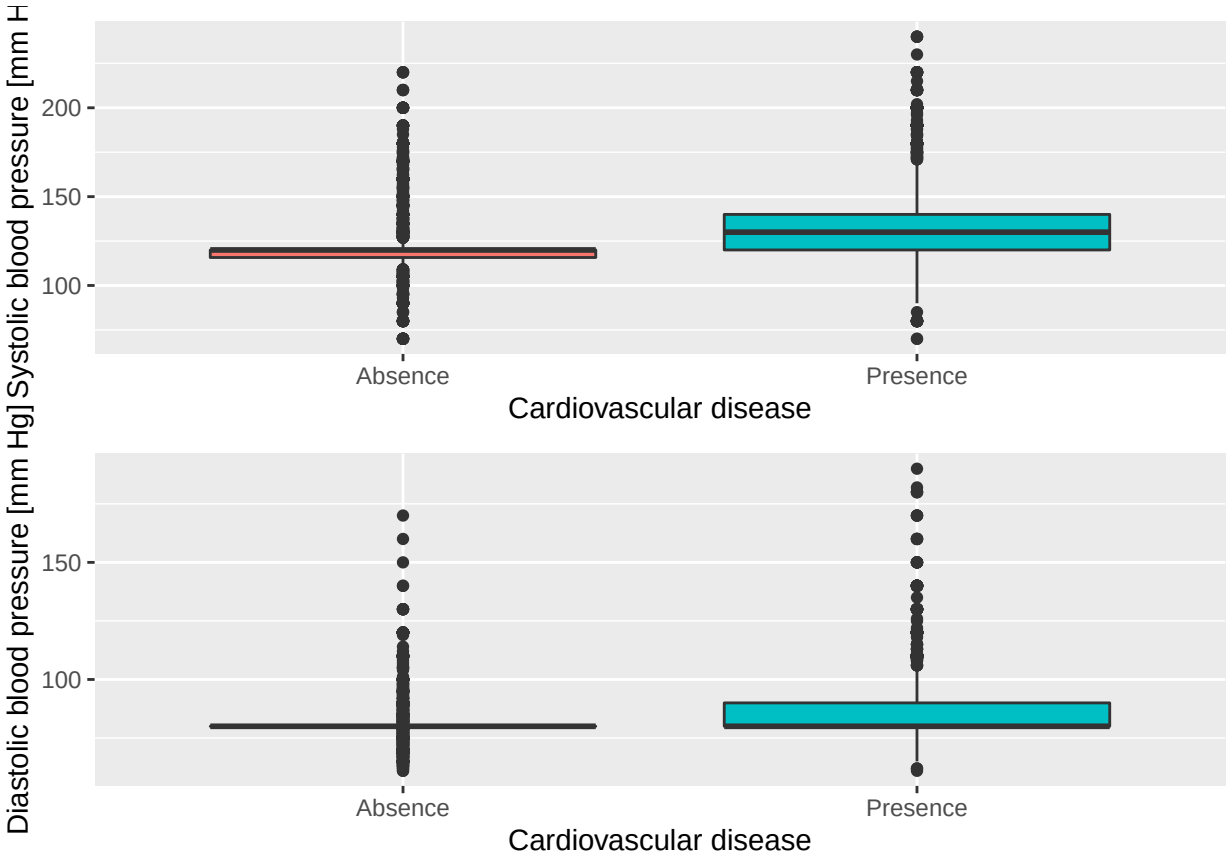

Fig. 1.7. Boxplot systolic blood pressure vs. diastolic blood pressure with cardiovascular disease status

```
mean(training_set[training_set$cardio == 0,]$ap_lo)
```

```
## [1] 79.36311
```

```
mean(training_set[training_set$cardio == 1,]$ap_lo)
```

```
## [1] 85.21589
```

Deviation in both cases deviation for diastolic blood pressure is not simetrical, and in diastolic blood pressure there is no overlapping in intervals but both cases contains lot of outliners.

Cholesterole can by high when person eats to much fatty food, it can be also genetically inherited. Cardiovascular disease in groups of level of cholesterol is shown i figure 1.8. Cardiovascular disease in present in all 3 group of levels of cholesterole. But it share in total count grows with level of cholesterole.

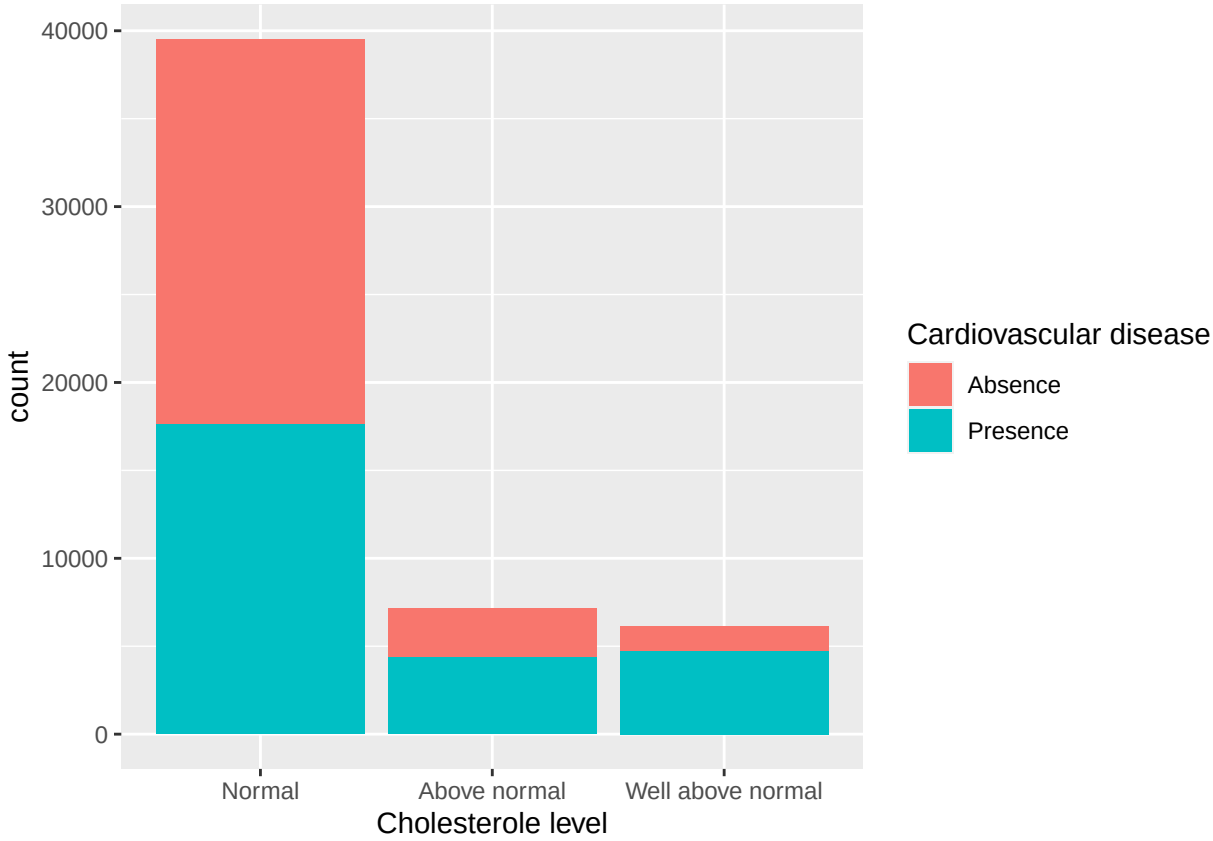

Fig. 1.8. Cardiovascular disease in groups of level of cholesterol

Similar behavior we can see with glucose level, it was shown in figure 1.9

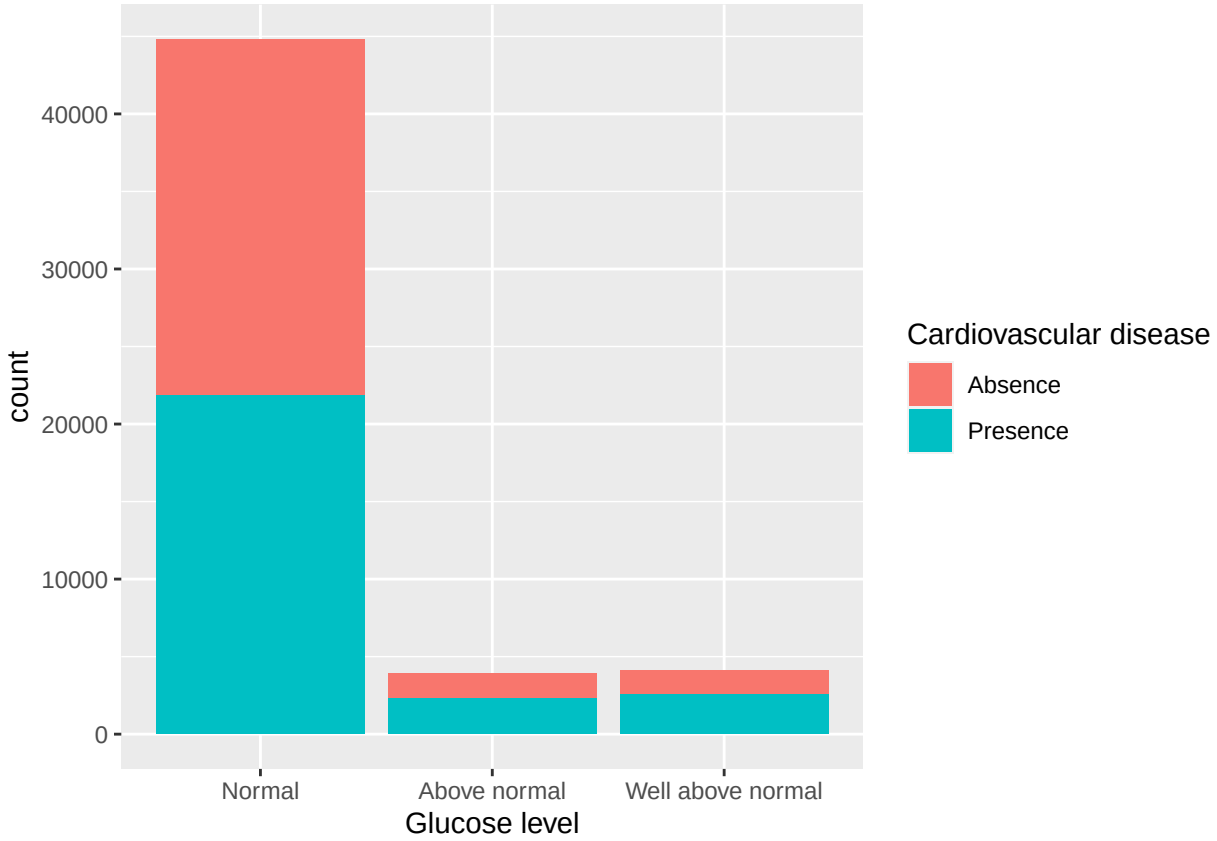

Figure. 1.9. Cardiovascular disease in groups of level of glucose

Smoking, alcohol intake and even physical activity have small influence on presence of cardiovascular disease, it is shown in figure 1.10.

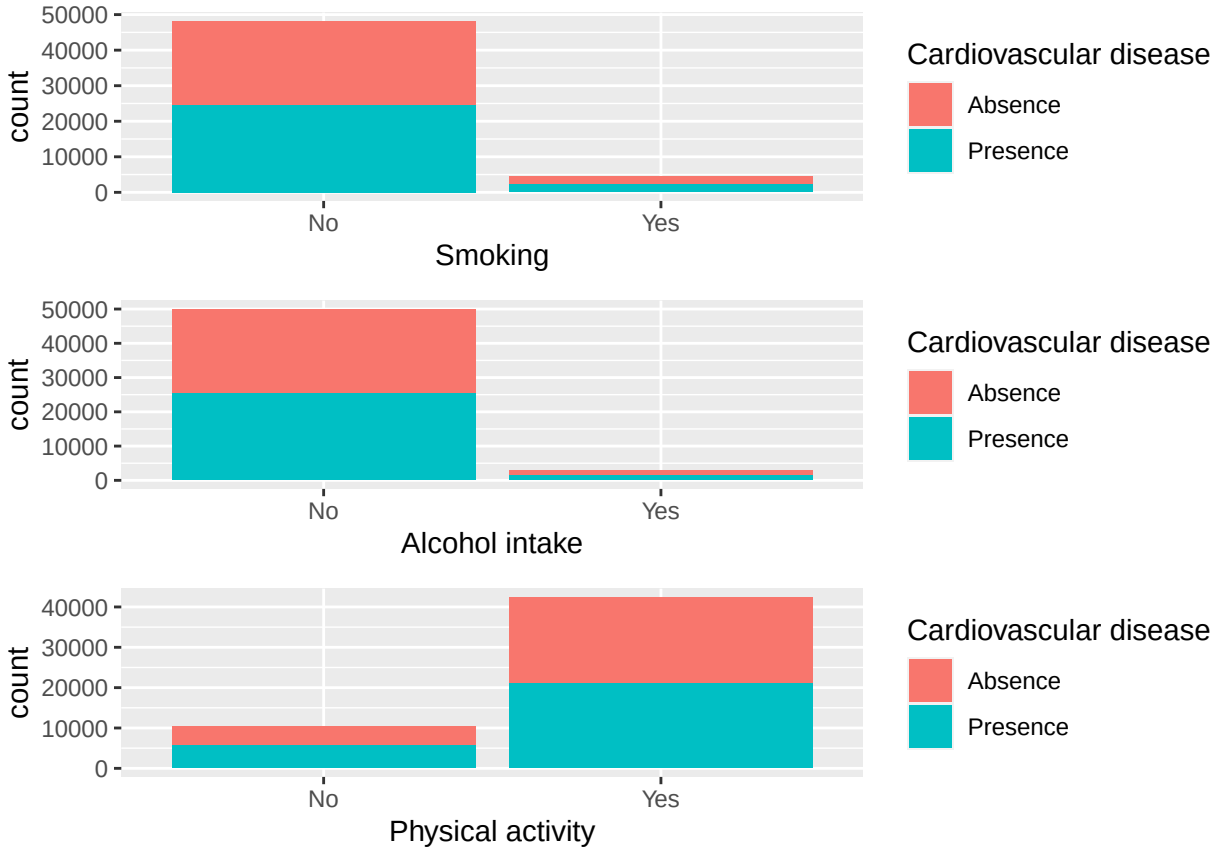

Figure. 1.10. Smoking, alcohol intake and physical activity influence on cardiovascular disease presence or absence

As we can see from figure 1.11, in our dataset, there is no strong corelation between values. We can see some weak correlation between ap_hi and cardio, and ap_lo and ap_hi.

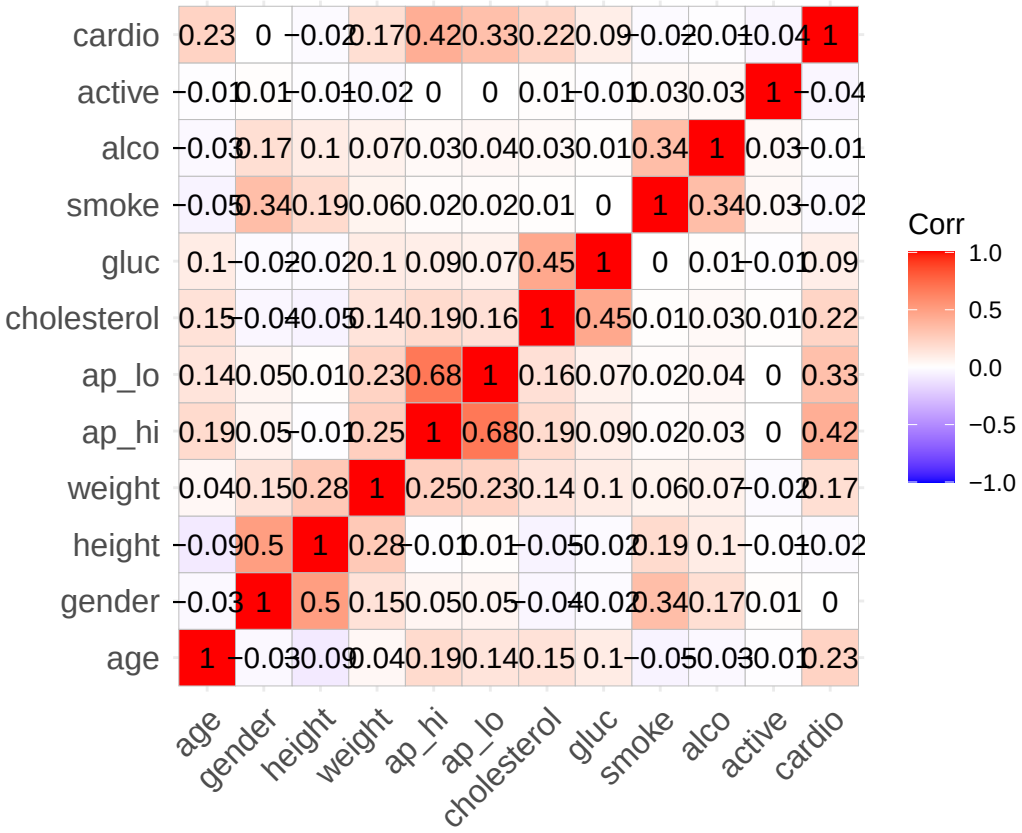

Fig. 1.11. Correlation betwwen values in dataset

2. Model creation and evaluate

For purpose of predicting abence of presence of cardiovacsular desease, we used 4 types of approach: - Neural network; - Random Forest - Logistic regression; - Logistic regression with regularization; - XGBoost.

Firstly, let's define our evaluate function. We will use it to check accuracy of our models.

```
ev_matt <- function (act, pred){
  TP <- sum(act == 1 & pred == 1)
  TN <- sum(act == 0 & pred == 0)
  FP <- sum(act == 0 & pred == 1)
  FN <- sum(act == 1 & pred == 0)

  denom <- as.double(TP+FP)*(TP+FN)*(TN+FP)*(TN+FN)
  if (any((TP+FP) == 0, (TP+FN) == 0, (TN+FP) == 0, (TN+FN) == 0)) denom <- 1
  mcc <- ((TP*TN)-(FP*FN)) / sqrt(denom)
  mcc
}

#Function for preprocess values:

mat_p <- function(val){
```

```
  round(val*100,1)
}
```

Our first model if a neural netowrk with 20 hidden neurons.

```
nn <- nnet(cardio~.,data=training_set,size=20,maxit=1000)
```

```
## # weights:  261
## initial  value 16901.398637
## iter  10 value 12152.225556
## iter  20 value 10891.796674
## iter  30 value 10485.402327
## iter  40 value 10317.338040
## iter  50 value 10215.149783
## iter  60 value 10142.637692
## iter  70 value 10045.123447
## iter  80 value 9841.742427
## iter  90 value 9733.013241
## iter 100 value 9696.158627
## iter 110 value 9662.051756
## iter 120 value 9645.956144
## iter 130 value 9639.463921
## iter 140 value 9633.120432
## iter 150 value 9628.730639
## iter 160 value 9626.359683
## iter 170 value 9624.552938
## iter 180 value 9622.921914
## iter 190 value 9620.885962
## iter 200 value 9619.920190
## iter 210 value 9617.216465
## iter 220 value 9615.485406
## iter 230 value 9614.784166
## iter 240 value 9614.554832
## iter 250 value 9614.335065
## iter 260 value 9614.130751
## iter 270 value 9613.809701
## iter 280 value 9613.387372
## iter 290 value 9613.088140
## iter 300 value 9612.874612
## iter 310 value 9612.611396
## iter 320 value 9612.174925
## iter 330 value 9611.738670
## iter 340 value 9611.545623
## iter 350 value 9611.350643
## iter 360 value 9611.162934
## iter 370 value 9611.078523
## iter 380 value 9610.980412
## iter 390 value 9610.816650
## iter 400 value 9610.607143
## iter 410 value 9610.444870
## iter 420 value 9610.354663
## iter 430 value 9610.269917
## iter 440 value 9610.071096
```

```
## iter 450 value 9609.954158
## iter 460 value 9609.867228
## iter 470 value 9609.754948
## iter 480 value 9609.645947
## iter 490 value 9609.551170
## iter 500 value 9609.449653
## iter 510 value 9609.402927
## iter 520 value 9609.366361
## iter 530 value 9609.318834
## iter 540 value 9609.297039
## iter 550 value 9609.278487
## iter 560 value 9609.249824
## final  value 9609.240034
## converged
```

```
preds = predict(nn,newdata=validation_set)
nn_score <- ev_matt(validation_set$cardio,preds>0.5)
```

Accuracy of this model is: 45.3%

The next model is random forest model:

```
training_setR <- training_set %>% select(-cardio)
rf <- randomForest(training_setR,as.factor(training_set$cardio))
validation_setR <- validation_set %>% select(-cardio)
rf_score <- ev_matt(predict(rf,newdata=validation_setR),validation_set$cardio)
```

Accuracy of this model is: 45%

Logistic regression:

```
logi <- glm(cardio~.,data=training_set,family=binomial(link="logit"))
logi_score <- ev_matt(predict(logi,type="response",newdata=validation_set)>0.5,
validation_set$cardio)
```

Accuracy of this model is: 44.2%

Logistic regression with regularization:

```
train.vars <- as.matrix(select(training_set,-cardio))
valid.vars <- as.matrix(select(validation_set,-cardio))
lgwr <- cv.glmnet(train.vars,training_set$cardio,family="binomial")
lgwr_score <- ev_matt(predict(lgwr,valid.vars,s="lambda.min",type="response")>0.5,
validation_set$cardio)
```

Accuracy of this model is: 44.4%

XGBoost:

```
xgbo <- xgboost(data=train.vars,label=training_set$cardio,nrounds=100)
```

```
## [1]  train-rmse:0.463881
## [2]  train-rmse:0.444544
```

```
## [3]  train-rmse:0.434235
## [4]  train-rmse:0.428719
## [5]  train-rmse:0.425562
## [6]  train-rmse:0.423630
## [7]  train-rmse:0.422476
## [8]  train-rmse:0.421668
## [9]  train-rmse:0.421012
## [10] train-rmse:0.420374
## [11] train-rmse:0.419834
## [12] train-rmse:0.419444
## [13] train-rmse:0.419108
## [14] train-rmse:0.418660
## [15] train-rmse:0.418213
## [16] train-rmse:0.417758
## [17] train-rmse:0.417475
## [18] train-rmse:0.417422
## [19] train-rmse:0.417172
## [20] train-rmse:0.416965
## [21] train-rmse:0.416735
## [22] train-rmse:0.416443
## [23] train-rmse:0.416138
## [24] train-rmse:0.415934
## [25] train-rmse:0.415776
## [26] train-rmse:0.415668
## [27] train-rmse:0.415412
## [28] train-rmse:0.415295
## [29] train-rmse:0.414924
## [30] train-rmse:0.414538
## [31] train-rmse:0.414339
## [32] train-rmse:0.414265
## [33] train-rmse:0.413972
## [34] train-rmse:0.413695
## [35] train-rmse:0.413560
## [36] train-rmse:0.413199
## [37] train-rmse:0.413071
## [38] train-rmse:0.412836
## [39] train-rmse:0.412521
## [40] train-rmse:0.412335
## [41] train-rmse:0.412312
## [42] train-rmse:0.411915
## [43] train-rmse:0.411813
## [44] train-rmse:0.411763
## [45] train-rmse:0.411566
## [46] train-rmse:0.411371
## [47] train-rmse:0.411364
## [48] train-rmse:0.411188
## [49] train-rmse:0.410927
## [50] train-rmse:0.410715
## [51] train-rmse:0.410537
## [52] train-rmse:0.410314
## [53] train-rmse:0.410179
## [54] train-rmse:0.409782
## [55] train-rmse:0.409432
## [56] train-rmse:0.409273
```

```
## [57] train-rmse:0.409126
## [58] train-rmse:0.408730
## [59] train-rmse:0.408637
## [60] train-rmse:0.408370
## [61] train-rmse:0.408218
## [62] train-rmse:0.408074
## [63] train-rmse:0.407902
## [64] train-rmse:0.407587
## [65] train-rmse:0.407225
## [66] train-rmse:0.407075
## [67] train-rmse:0.407046
## [68] train-rmse:0.406713
## [69] train-rmse:0.406565
## [70] train-rmse:0.406403
## [71] train-rmse:0.406269
## [72] train-rmse:0.405958
## [73] train-rmse:0.405770
## [74] train-rmse:0.405705
## [75] train-rmse:0.405698
## [76] train-rmse:0.405550
## [77] train-rmse:0.405528
## [78] train-rmse:0.405364
## [79] train-rmse:0.405326
## [80] train-rmse:0.405256
## [81] train-rmse:0.405212
## [82] train-rmse:0.405009
## [83] train-rmse:0.404786
## [84] train-rmse:0.404511
## [85] train-rmse:0.404317
## [86] train-rmse:0.404271
## [87] train-rmse:0.404098
## [88] train-rmse:0.403901
## [89] train-rmse:0.403775
## [90] train-rmse:0.403583
## [91] train-rmse:0.403427
## [92] train-rmse:0.403345
## [93] train-rmse:0.403127
## [94] train-rmse:0.402777
## [95] train-rmse:0.402575
## [96] train-rmse:0.402375
## [97] train-rmse:0.402288
## [98] train-rmse:0.402246
## [99] train-rmse:0.402208
## [100]    train-rmse:0.401937
```

```
xgbo_score <- ev_matt(predict(xgbo,newdata=valid.vars)>0.5,validation_set$cardio)
```

Accuracy of this model is: 45.7%

2.1. Results

Accuracy of each of model was quite similar, it can mean that the value we want to predict is not dependant on features that we have in out dataset. Overall the best model was a random tree model.

Conclusion

The best way to improve models accuracy is to find relation between something and the presence of cardiovascular desease and colect information about it to check accuracy of models on the new data. The other case can be it that data is acquired in messy way, as we could see earlier in preprocessing part.