Exercise 9

Egzon Syka and Labinot Jakupi

1.

First we load the data and show a summary of it:

```
library(boot)

data <- read.table("DataHeart.txt", header=T)

summary(data)
```

```
##        ID              age              sex              pain
##  Min.   :  1.00   Min.   : 3.00   Min.   :0.0000   Min.   :1.000
##  1st Qu.: 68.75   1st Qu.:47.75   1st Qu.:0.0000   1st Qu.:3.000
##  Median :136.50   Median :55.00   Median :1.0000   Median :3.000
##  Mean   :136.50   Mean   :54.24   Mean   :0.6765   Mean   :3.173
##  3rd Qu.:204.25   3rd Qu.:61.00   3rd Qu.:1.0000   3rd Qu.:4.000
##  Max.   :272.00   Max.   :77.00   Max.   :1.0000   Max.   :4.000
##       pres        cholesterol        sugar           electro
##  Min.   : 94.0   Min.   :125.0   Min.   :0.0000   Min.   :0.000
##  1st Qu.:120.0   1st Qu.:212.8   1st Qu.:0.0000   1st Qu.:0.000
##  Median :130.0   Median :245.0   Median :0.0000   Median :2.000
##  Mean   :131.3   Mean   :249.3   Mean   :0.1471   Mean   :1.029
##  3rd Qu.:140.0   3rd Qu.:278.0   3rd Qu.:0.0000   3rd Qu.:2.000
##  Max.   :200.0   Max.   :564.0   Max.   :1.0000   Max.   :2.000
##    gramstein           rate           angina           fiss
##  Min.   :-4.500   Min.   : 71.0   Min.   :0.0000   Min.   :11.00
##  1st Qu.: 9.300   1st Qu.:132.8   1st Qu.:0.0000   1st Qu.:22.00
##  Median :10.100   Median :153.5   Median :0.0000   Median :25.00
##  Mean   : 9.975   Mean   :149.6   Mean   :0.3346   Mean   :24.94
##  3rd Qu.:10.700   3rd Qu.:166.0   3rd Qu.:1.0000   3rd Qu.:28.00
##  Max.   :13.300   Max.   :202.0   Max.   :1.0000   Max.   :39.00
##       peak          slope          vessels            thal
##  Min.   :0.00   Min.   :1.000   Min.   :0.0000   Min.   :3.000
##  1st Qu.:0.00   1st Qu.:1.000   1st Qu.:0.0000   1st Qu.:3.000
##  Median :0.80   Median :2.000   Median :0.0000   Median :3.000
##  Mean   :1.05   Mean   :1.588   Mean   :0.6765   Mean   :4.713
##  3rd Qu.:1.65   3rd Qu.:2.000   3rd Qu.:1.0000   3rd Qu.:7.000
##  Max.   :6.20   Max.   :3.000   Max.   :3.0000   Max.   :7.000
##       blst          disease
##  Min.   :50.14   Min.   :1.000
##  1st Qu.:57.50   1st Qu.:1.000
##  Median :66.01   Median :1.000
##  Mean   :65.28   Mean   :1.449
##  3rd Qu.:71.88   3rd Qu.:2.000
##  Max.   :79.77   Max.   :2.000
```

Then we see that there are some invalid data(like the age 3 when it's supposed to be in range 20-80) we need to remove them, then we also see that there are some categorical values that we know from the description of the data, we must convert them to factor:

```
data <- data[-1]
data <- data[which(data$age >= 20),]
#data <- data[which(data$gramstein > 6),]

## will use this just for correleation - non numeric data needed
temp <- data

## change categorical data to factor
data$sex <- as.factor(data$sex)
data$pain <- as.factor(data$pain)
data$disease <- as.factor(data$disease)
data$electro <- as.factor(data$electro)
data$vessels <- as.factor(data$vessels)
data$thal <- as.factor(data$thal)
data$sugar <- as.factor(data$sugar)

summary(data)
```

```
##       age        sex     pain         pres        cholesterol    sugar
##  Min.   :29.00   0: 88   1: 20   Min.   : 94.0   Min.   :126.0   0:231
##  1st Qu.:48.00   1:183   2: 43   1st Qu.:120.0   1st Qu.:213.0   1: 40
##  Median :55.00           3: 79   Median :130.0   Median :245.0
##  Mean   :54.43           4:129   Mean   :131.3   Mean   :249.7
##  3rd Qu.:61.00                   3rd Qu.:140.0   3rd Qu.:279.0
##  Max.   :77.00                   Max.   :200.0   Max.   :564.0
##  electro   gramstein          rate           angina           fiss
##  0:131   Min.   :-4.500   Min.   : 71.0   Min.   :0.0000   Min.   :11.00
##  1:  2   1st Qu.: 9.300   1st Qu.:132.5   1st Qu.:0.0000   1st Qu.:22.00
##  2:138   Median :10.100   Median :153.0   Median :0.0000   Median :25.00
##          Mean   : 9.975   Mean   :149.5   Mean   :0.3321   Mean   :24.98
##          3rd Qu.:10.700   3rd Qu.:166.0   3rd Qu.:1.0000   3rd Qu.:28.00
##          Max.   :13.300   Max.   :202.0   Max.   :1.0000   Max.   :39.00
##       peak           slope       vessels thal         blst       disease
##  Min.   :0.000   Min.   :1.000   0:160   3:152   Min.   :50.14   1:150
##  1st Qu.:0.000   1st Qu.:1.000   1: 59   6: 14   1st Qu.:57.66   2:121
##  Median :0.800   Median :2.000   2: 33   7:105   Median :66.03
##  Mean   :1.054   Mean   :1.587   3: 19           Mean   :65.32
##  3rd Qu.:1.700   3rd Qu.:2.000                   3rd Qu.:71.92
##  Max.   :6.200   Max.   :3.000                   Max.   :79.77
```

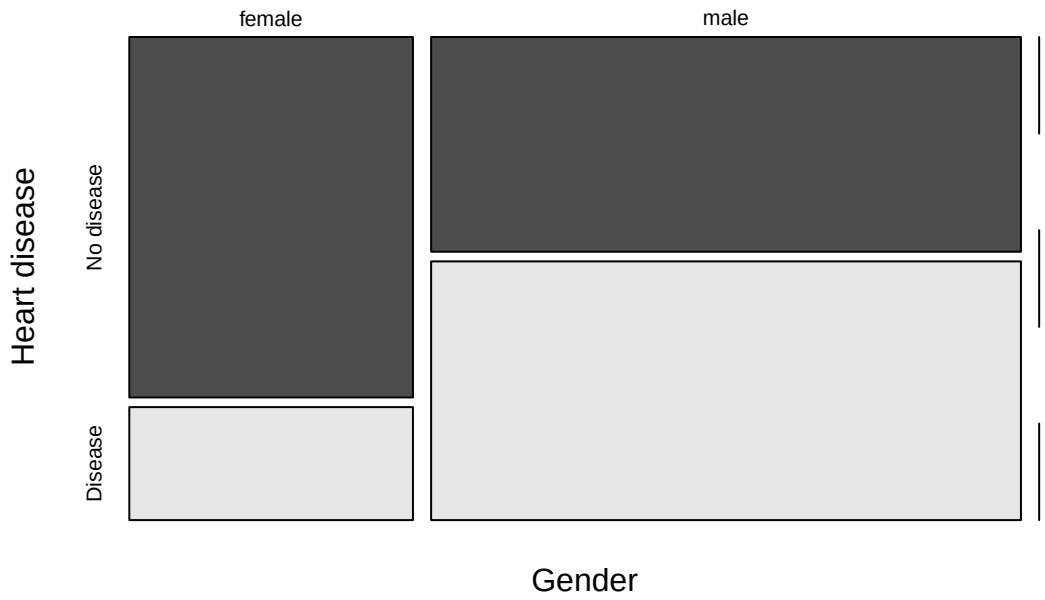
We try to check the pairwise corelation table between variables(features):

```
## trying to find which values are not so important - to reduce overfitting(noise)

# cor(temp)
```

we notice that there is some corelation between the **peak** and **slope** variables for example. Now, we will just try to ilustrate the effect of for example of *sex* variable on *disease*:

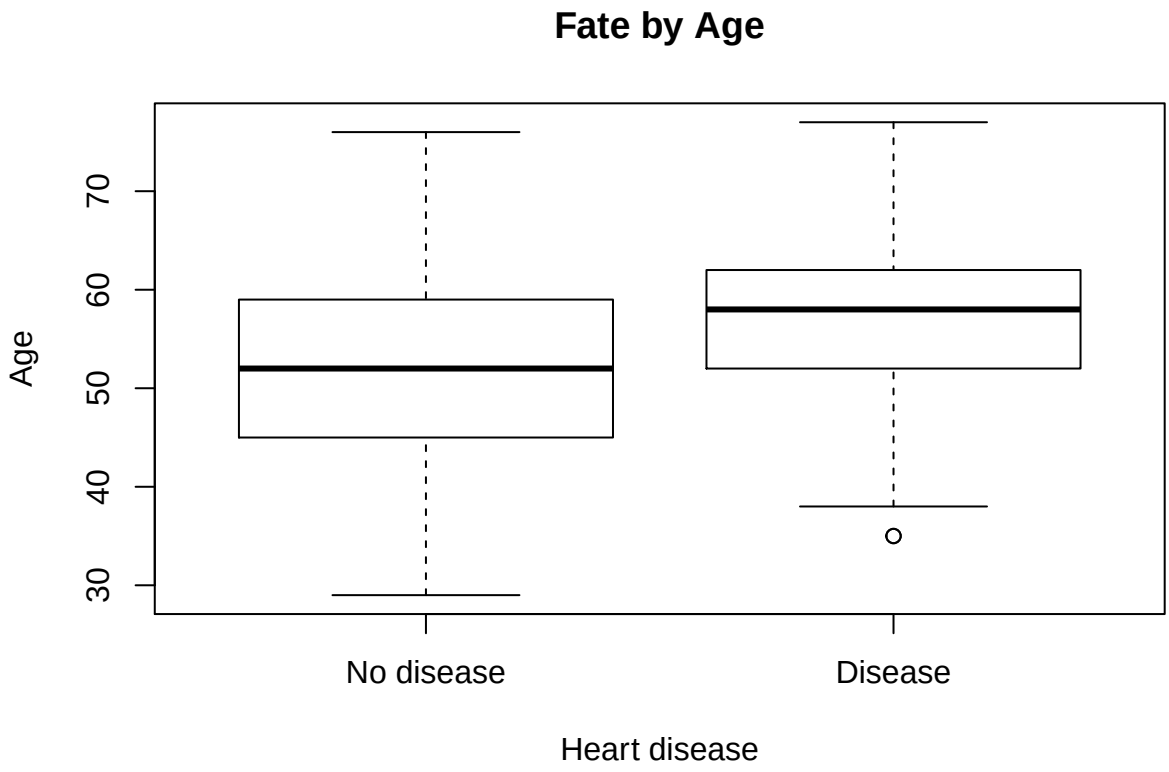
```
heart = data
levels(heart$disease) = c("No disease","Disease")
levels(heart$sex) = c("female","male","")
mosaicplot(heart$sex ~ heart$disease,
           main="Fate by Sex", shade=FALSE,color=TRUE,
           xlab="Gender", ylab="Heart disease")
```



we clearly see from the graph above that there is more risk for mens to get heart disease.

Same we can also do for the age:

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



Now, we start building different models, here we start with a logistic regression using the complete

dataset(except ID var.) and then we remove the *slope* and *gramstein* since them don't contribute much and slope is also correlated to *peak* variable:

```
set.seed(7)
## building models - logistic

#model full set
logistic.mod <- glm( disease~., data, family =binomial )
cv.error <- cv.glm(data, logistic.mod, K=10)$delta[1]

# remove slope because covariance with peak is high
newData <- subset(data, select=-c(slope, gramstein))

#model without slope and gramstein
logistic.mod1 <- glm( disease~., newData, family = binomial )
cv.error1 <- cv.glm(newData, logistic.mod1)$delta[1]
```

We normalize the data in order to see if normalization can increase the accuracy:

```
set.seed(7)

## trying with scaled data

numdata <- read.table("DataHeart.txt", header=T)
numdata <- numdata[-1]
numdata <- numdata[which(numdata$age >= 20),]
numdata <- numdata[which(numdata$gramstein > 6),]
disease <- as.factor(numdata$disease)

sData <- as.data.frame(scale(subset(numdata, select = - disease)))
sData <- cbind(sData, disease)

logistic.mod2 <- glm( disease ~ sex + pain + pres + electro + rate + angina +
    peak + vessels + thal, data = sData, family=binomial)
cv.error2 <- cv.glm(sData, logistic.mod2, K=10)$delta[1]

newData <- subset(sData, select=-c(slope, gramstein))

logistic.mod3 <- glm( disease~., newData, family = binomial )
cv.error3 <- cv.glm(sData, logistic.mod3)$delta[1]
```

it appears that it doesn't matter a lot, but we noticed is that the model where we exclude some variables like *sugar*, *cholesterol*, *slope*, *gramstein*,*fiss*,*blst* it performs better than the other models.

So, let's try to check the importance of variables by using the *summary* function and checking for variables that have lowest *p-values*:

```
summary(logistic.mod)
```

```
##
## Call:
## glm(formula = disease ~ ., family = binomial, data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8078  -0.4800  -0.1093   0.3006   2.9115
##
```

```
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -10.601254   4.144656  -2.558 0.010533 *
## age          -0.026342   0.027102  -0.972 0.331068
## sex1          1.688225   0.607724   2.778 0.005470 **
## pain2         1.624471   0.924701   1.757 0.078960 .
## pain3         0.743044   0.791287   0.939 0.347714
## pain4         2.618453   0.814468   3.215 0.001305 **
## pres          0.030112   0.012693   2.372 0.017676 *
## cholesterol   0.007386   0.004354   1.696 0.089796 .
## sugar1       -0.348930   0.612092  -0.570 0.568636
## electro1      0.997176   3.028372   0.329 0.741946
## electro2      0.634997   0.434813   1.460 0.144183
## gramstein     0.120059   0.147259   0.815 0.414906
## rate         -0.025485   0.012162  -2.096 0.036126 *
## angina        0.659136   0.472394   1.395 0.162923
## fiss          0.030045   0.041160   0.730 0.465418
## peak          0.457856   0.260679   1.756 0.079021 .
## slope         0.626444   0.421035   1.488 0.136786
## vessels1      2.123312   0.540190   3.931 8.47e-05 ***
## vessels2      3.153635   0.807079   3.907 9.33e-05 ***
## vessels3      2.045940   0.927344   2.206 0.027368 *
## thal6        -0.056845   0.888375  -0.064 0.948980
## thal7         1.557674   0.469923   3.315 0.000917 ***
## blst          0.021880   0.025247   0.867 0.386136
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 372.58  on 270  degrees of freedom
## Residual deviance: 163.17  on 248  degrees of freedom
## AIC: 209.17
##
## Number of Fisher Scoring iterations: 6
```

Now let's experiment with different variables:

```
set.seed(10)

logistic.mod4 <- glm(disease ~ thal + vessels + pain + sex + pres + peak + slope + angina + rate + age,
cv.error4 <- cv.glm(data, logistic.mod4, K=10)$delta[1]

logistic.mod5 <- glm(disease ~ thal + pain + vessels + rate + peak + angina + sex + age + slope + blst,
cv.error5 <- cv.glm(data, logistic.mod5, K=10)$delta[1]
```

We use **mlbench** and **caret** libraries to measure the importance of variables(within LDA model):

```
set.seed(7)
# load the library
library(mlbench)
library(caret)
```

```
## Loading required package: lattice
```

```
##
```

```
## Attaching package: 'lattice'
```

```
## The following object is masked from 'package:boot':
##
##     melanoma
```

```
## Loading required package: ggplot2
```

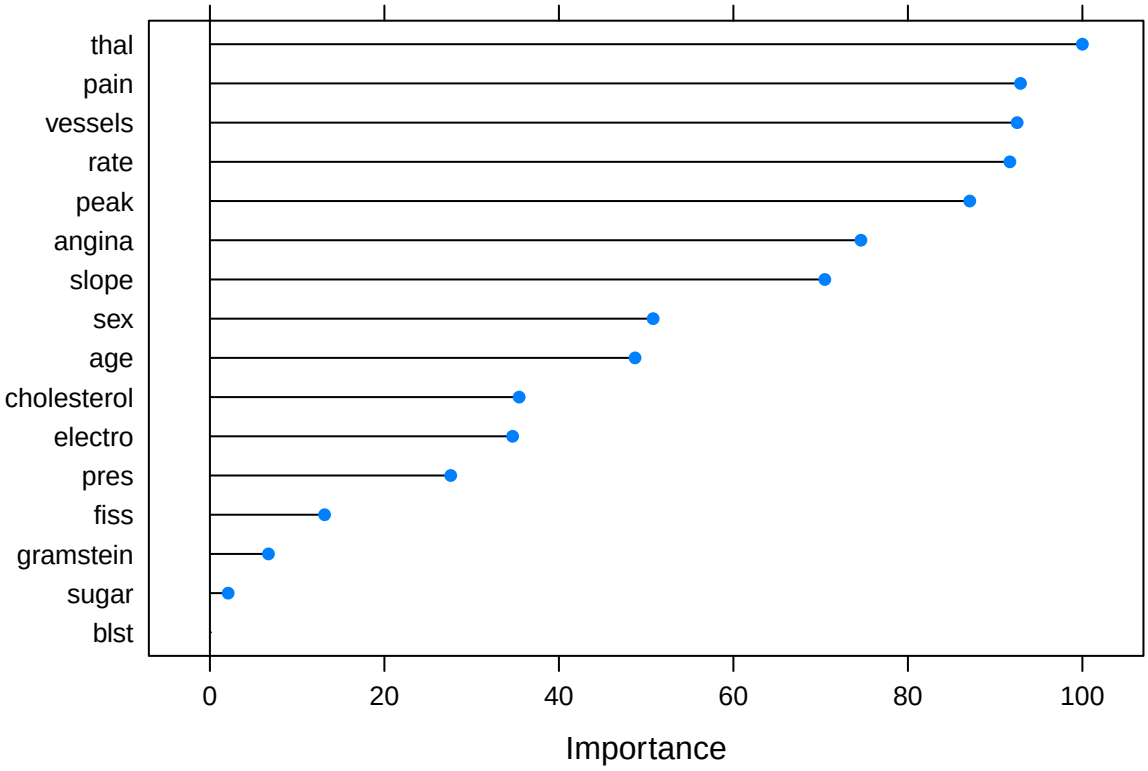
```
# prepare training scheme
control <- trainControl(method="repeatedcv", number=10, repeats=3)
# train the model
model <- train(disease~., data=data, method="lda", preProcess="scale", trControl=control)
```

```
## Loading required package: MASS
```

```
# estimate variable importance
importance <- varImp(model, scale=TRUE)
# summarize importance
print(importance)
```

```
## ROC curve variable importance
##
##             Importance
## thal           100.000
## pain            92.906
## vessels         92.525
## rate            91.690
## peak            87.091
## angina          74.616
## slope           70.481
## sex             50.799
## age             48.727
## cholesterol     35.447
## electro         34.694
## pres            27.601
## fiss            13.146
## gramstein        6.712
## sugar            2.093
## blst             0.000
```

```
# plot importance
plot(importance)
```



We want to select features that have at least 40% importancem but we can clearly see that *LDA* doesn't do a better job than *glm*:

```
## LDA model
set.seed(7)
library(MASS)

heart <- subset(data, select = -c(fiss, gramstein, sugar, blst, cholesterol, electro))

lda.mod <- lda(disease ~ ., heart, CV=T)
cat("\nLDA Test Error rate = ", mean(lda.mod$class != heart$disease))
```

```
##
## LDA Test Error rate =  0.1476015
```

```
lda.mod1 <- lda(disease ~ ., newData, CV=T)
cat("\nLDA Test Error rate = ", mean(lda.mod1$class != newData$disease))
```

```
##
## LDA Test Error rate =  0.1703704
```

We also tried to use randomForest to check the accuracy that we could get and with how many variables we get the best result(it seems that with 8 variables we get the best):

```
set.seed(7)
# load the library
library(mlbench)
library(caret)
# define the control using a random forest selection function
control <- rfeControl(functions=rfFuncs, method="cv", number=10)
# run the RFE algorithm
results <- rfe(data[-17], data$disease, sizes=c(1:16), rfeControl=control)
```

```
## Loading required package: randomForest
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
## The following object is masked from 'package:ggplot2':
##
##     margin
```

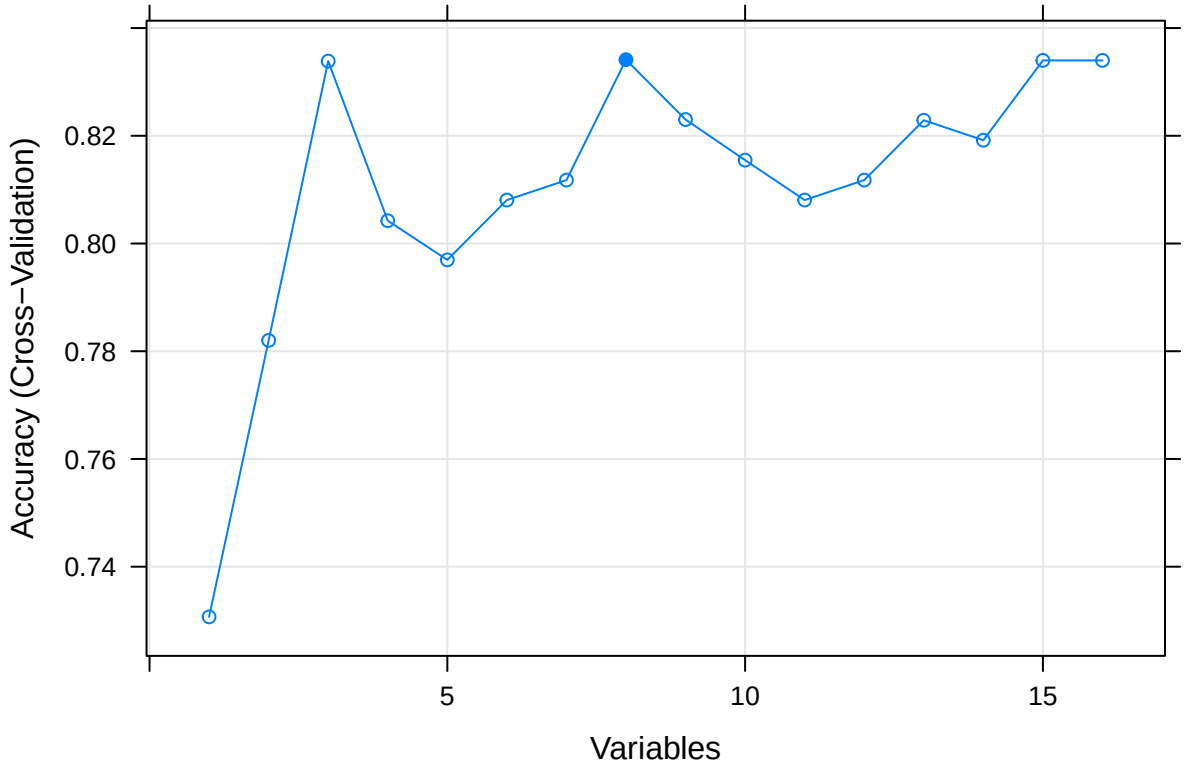
```
# summarize the results
print(results)
```

```
##
## Recursive feature selection
##
## Outer resampling method: Cross-Validated (10 fold)
##
## Resampling performance over subset size:
##
##  Variables Accuracy  Kappa AccuracySD KappaSD Selected
##          1   0.7307 0.4523    0.09545 0.19415
##          2   0.7820 0.5571    0.08324 0.16588
##          3   0.8339 0.6613    0.06836 0.14165
##          4   0.8042 0.6005    0.06110 0.12473
##          5   0.7970 0.5853    0.05340 0.11342
##          6   0.8081 0.6082    0.06259 0.13186
##          7   0.8118 0.6172    0.04445 0.08942
##          8   0.8341 0.6627    0.06521 0.13198        *
##          9   0.8230 0.6405    0.04473 0.08918
##         10   0.8155 0.6259    0.05794 0.11273
##         11   0.8081 0.6105    0.05751 0.11315
##         12   0.8118 0.6189    0.05653 0.11072
##         13   0.8229 0.6386    0.03817 0.07838
##         14   0.8192 0.6321    0.04075 0.07943
##         15   0.8340 0.6616    0.03111 0.06018
##         16   0.8340 0.6626    0.04677 0.09337
##
## The top 5 variables (out of 8):
##    thal, vessels, pain, peak, rate
```

```
# list the chosen features
predictors(results)
```

```
## [1] "thal"    "vessels" "pain"    "peak"    "rate"    "slope"   "sex"
## [8] "angina"
```

```
# plot the results
plot(results, type=c("g", "o"))
```



We also tried the **SVM** model with different kernels (polynomial, linear and rbf) but the results weren't promising compared to those of glm:

```
## trying with svm, with differnet kernels - but until now logistic regression is the best

library(e1071)
set.seed(7)
tune.out <- tune(svm, disease~., data = data, kernel = 'polynomial',
                 ranges=list(cost=c(0.0001,0.001,0.01,0.1,1,5,10), degree=c(2,3,4,5)))
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost degree
##     5      2
##
## - best performance: 0.1993386
##
## - Detailed performance results:
##     cost degree     error dispersion
## 1  1e-04      2 0.4468254 0.10657031
## 2  1e-03      2 0.4468254 0.10657031
## 3  1e-02      2 0.4468254 0.10657031
## 4  1e-01      2 0.4394180 0.10646811
## 5  1e+00      2 0.2400794 0.11651104
## 6  5e+00      2 0.1993386 0.07465978
```

```
## 7  1e+01      2 0.2177249 0.08454624
## 8  1e-04      3 0.4468254 0.10657031
## 9  1e-03      3 0.4468254 0.10657031
## 10 1e-02      3 0.4468254 0.10657031
## 11 1e-01      3 0.4394180 0.10646811
## 12 1e+00      3 0.2843915 0.10824133
## 13 5e+00      3 0.2439153 0.09495844
## 14 1e+01      3 0.2326720 0.08244195
## 15 1e-04      4 0.4468254 0.10657031
## 16 1e-03      4 0.4468254 0.10657031
## 17 1e-02      4 0.4468254 0.10657031
## 18 1e-01      4 0.4431217 0.11418476
## 19 1e+00      4 0.3802910 0.09158189
## 20 5e+00      4 0.2734127 0.11130699
## 21 1e+01      4 0.2661376 0.10070068
## 22 1e-04      5 0.4468254 0.10657031
## 23 1e-03      5 0.4468254 0.10657031
## 24 1e-02      5 0.4468254 0.10657031
## 25 1e-01      5 0.4468254 0.10657031
## 26 1e+00      5 0.4210317 0.09241616
## 27 5e+00      5 0.3358466 0.09160566
## 28 1e+01      5 0.2953704 0.09607148
```

```
tune.out1 <- tune(svm, disease~thal + pain + vessels + rate + peak + angina + sex + age + slope, data =
                  ranges=list(cost=c(0.00001,0.0001,0.001,0.01,0.1,1,5,10)))
#summary(tune.out1)
```

We also give a try to a deep learning model(neural network) with 6 hidden layers with 10 nodes in each of them but the overall prediction accuracy was the same:

```
inTraining <- createDataPartition(data$disease, p = 0.70, list = FALSE)
training <- data[ inTraining, ]
testing  <- data[-inTraining, ]

require(h2o);
```

```
## Loading required package: h2o
```

```
##
## ----------------------------------------------------------------------
##
## Your next step is to start H2O:
##     > h2o.init()
##
## For H2O package documentation, ask for help:
##     > ??h2o
##
## After starting H2O, you can use the Web UI at http://localhost:54321
## For more information visit http://docs.h2o.ai
##
## ----------------------------------------------------------------------
```

```
##
## Attaching package: 'h2o'
```

```
## The following objects are masked from 'package:stats':
##
```

```
##     cor, sd, var

## The following objects are masked from 'package:base':
##
##     ||, &&, %*%, apply, as.factor, as.numeric, colnames,
##     colnames<-, ifelse, %in%, is.character, is.factor, is.numeric,
##     log, log10, log1p, log2, round, signif, trunc
```

```
h2o.no_progress()
localH2O <-h2o.init()
```

```
##  Connection successful!
##
## R is connected to the H2O cluster:
##     H2O cluster uptime:         50 minutes 48 seconds
##     H2O cluster version:        3.10.4.6
##     H2O cluster version age:    1 month and 4 days
##     H2O cluster name:           H2O_started_from_R_syka_dej860
##     H2O cluster total nodes:    1
##     H2O cluster total memory:   1.62 GB
##     H2O cluster total cores:    4
##     H2O cluster allowed cores:  2
##     H2O cluster healthy:        TRUE
##     H2O Connection ip:          localhost
##     H2O Connection port:        54321
##     H2O Connection proxy:       NA
##     H2O Internal Security:      FALSE
##     R Version:                  R version 3.4.0 (2017-04-21)
```

```
heart.train <- as.h2o(training)
heart.test <- as.h2o(testing)
model <- h2o.deeplearning( x = setdiff(colnames(heart.train),c("disease")),
                           y = 'disease',
                           training_frame = heart.train,
                           activation = "RectifierWithDropout",
                           hidden = c(10, 10, 10, 10, 10, 10),
                           epochs = 100000 )
predictions <- h2o.predict(model, heart.test)

suppressMessages(require(ROCR, quietly = T))
preds <- as.data.frame(predictions)
labels <- as.data.frame(heart.test$disease)

confusionMatrix(preds$predict, testing$disease)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2
##          1 39 13
##          2  6 23
##
##                Accuracy : 0.7654
##                  95% CI : (0.6582, 0.8525)
##     No Information Rate : 0.5556
##     P-Value [Acc > NIR] : 7.211e-05
```

```
##
##                  Kappa : 0.5156
##  Mcnemar's Test P-Value : 0.1687
##
##             Sensitivity : 0.8667
##             Specificity : 0.6389
##          Pos Pred Value : 0.7500
##          Neg Pred Value : 0.7931
##              Prevalence : 0.5556
##          Detection Rate : 0.4815
##    Detection Prevalence : 0.6420
##       Balanced Accuracy : 0.7528
##
##        'Positive' Class : 1
##
```

2.

```
set.seed(97)
#calculating the cost of wrong prediction
library(caret)

inTraining <- createDataPartition(data$disease, p = 0.7, list = FALSE)
training <- data[ inTraining, ]
testing  <- data[-inTraining, ]


model <- glm(disease ~ thal + vessels + pain + sex + pres + peak + slope + angina + rate
                + age, training, family = binomial)

predictions <- predict(model, testing)

fitpredt <- function(t) ifelse(predictions > t , 2, 1)

confMatrix <- confusionMatrix(fitpredt(0.76), testing$disease)

confMatrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2
##          1 41  9
##          2  4 27
##
##                Accuracy : 0.8395
##                  95% CI : (0.7412, 0.9117)
##     No Information Rate : 0.5556
##     P-Value [Acc > NIR] : 5.633e-08
##
##                   Kappa : 0.6704
##  Mcnemar's Test P-Value : 0.2673
```

```
##
##             Sensitivity : 0.9111
##             Specificity : 0.7500
##          Pos Pred Value : 0.8200
##          Neg Pred Value : 0.8710
##              Prevalence : 0.5556
##          Detection Rate : 0.5062
##    Detection Prevalence : 0.6173
##       Balanced Accuracy : 0.8306
##
##        'Positive' Class : 1
##
```

```
false_absent <- confMatrix$table[1,2] #predicting absent when it's present, cost = 3

false_present <- confMatrix$table[2,1] #predicting present when it's absent, cost = 1

accuracy_on_false_absent <- false_absent / (confMatrix$table[1,1] + false_absent) #18 percent accuracy

accuracy_on_false_present <- false_present / (confMatrix$table[2,2] + false_present) # 12 percent

cost_of_misclassification <- false_absent * 3 + false_present
```

So our model predicts 'absent' when the disease is 'present' in 18% of cases and predicts 'present' when it's 'absent' in 12% of cases.

The deep learning model predicts 'absent' when the disease is 'present' in 13% of cases and predicts 'present' when it is 'absent' in 14% of cases.

So in this aspect that model is better because it minimizes the risk of not detecting the presence of disease.