Risk of the Life

Hüseyin KAYAR

4/8/23

Supervised Learning: Logistic Regression Model

In this application we will try to make predictions on heart attack data using logistic regression. Features and target of the problem are listed below.

Features ;

Age : Age of the patient Sex : Sex of the patient cp : Chest Pain type chest pain type exang: exercise induced angina (1 = yes; 0 = no) ca: number of major vessels (0-3) trtbps : resting blood pressure (in mm Hg) chol : cholestoral in mg/dl fetched via BMI sensor fbs : (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false) restecg : resting electrocardiographic results thalach : maximum heart rate achieved oldpeak : previous peak sip : slope thall : thal rate

Target;

output : 0= less chance of heart attack 1= more chance of heart attack

By using these feature we try to predict the target so, we will be able to predict under which conditions the risk of heart attack is higher.

Packages

We need some packages to be able to train the model, measure model performance and check over and under fitting problem. These are downloaded and invoked sequentially as listed below.

```r
install.packages("tidyverse")
install.packages("readr")
install.packages("caret")
install.packages("dplyr")
install.packages("DALEX")
library(tidyverse)
```

```r
library(readr)
library(caret)
library(DALEX)
library(dplyr)
```

Dataset

"Heart Attack Analysis & Prediction Data set" is used to train logistic regression model and make prediction of the heart attack risk of the people. Data set contains 14 variable. By using "read_csv()" code we load the data set.

```r
data <- read_csv("heart.csv")
```

Variable Types

To see variable types we can use "str()" code.

```r
str(data)
```

```
spc_tbl_ [303 x 14] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ age     : num [1:303] 63 37 41 56 57 57 56 44 52 57 ...
 $ sex     : num [1:303] 1 1 0 1 0 1 0 1 1 1 ...
 $ cp      : num [1:303] 3 2 1 1 0 0 1 1 2 2 ...
 $ trtbps  : num [1:303] 145 130 130 120 120 140 140 120 172 150 ...
 $ chol    : num [1:303] 233 250 204 236 354 192 294 263 199 168 ...
 $ fbs     : num [1:303] 1 0 0 0 0 0 0 0 1 0 ...
 $ restecg : num [1:303] 0 1 0 1 1 1 0 1 1 1 ...
 $ thalachh: num [1:303] 150 187 172 178 163 148 153 173 162 174 ...
 $ exng    : num [1:303] 0 0 0 0 1 0 0 0 0 0 ...
 $ oldpeak : num [1:303] 2.3 3.5 1.4 0.8 0.6 0.4 1.3 0 0.5 1.6 ...
 $ slp     : num [1:303] 0 0 2 2 2 1 1 2 2 2 ...
 $ caa     : num [1:303] 0 0 0 0 0 0 0 0 0 0 ...
 $ thall   : num [1:303] 1 2 2 2 2 1 2 3 3 2 ...
 $ output  : num [1:303] 1 1 1 1 1 1 1 1 1 1 ...
 - attr(*, "spec")=
  .. cols(
  ..   age = col_double(),
  ..   sex = col_double(),
```

```
  ..   cp = col_double(),
  ..   trtbps = col_double(),
  ..   chol = col_double(),
  ..   fbs = col_double(),
  ..   restecg = col_double(),
  ..   thalachh = col_double(),
  ..   exng = col_double(),
  ..   oldpeak = col_double(),
  ..   slp = col_double(),
  ..   caa = col_double(),
  ..   thall = col_double(),
  ..   output = col_double()
  .. )
 - attr(*, "problems")=<externalptr>
```

Dimension

```
dim(data)
```

```
[1] 303  14
```

There 303 entries in the data set. We should eliminate the missing data so our classification's success rate can be high.

```
data <- na.exclude(data)
```

Traning

During the training phase, train and test sets are used and these sets should contain different data from each other. I did this by randomly splitting the original data set.

As first step, I created a train set. I set the train set to contain 80% of the entire data set. In addition, as the second stage, I used 20% of the test data set.

```
set.seed(123)
index <- sample(1:nrow(data),round(nrow(data)*0.80))
train <- data[index, ]
test <- data[-index, ]
```

This is the first 10 rows in the train set:

```
train
```

```
# A tibble: 242 x 14
     age   sex    cp trtbps  chol   fbs restecg thalachh  exng oldpeak   slp
   <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>   <dbl>    <dbl> <dbl>   <dbl> <dbl>
 1    43     1     0    120   177     0       0      120     1     2.5     1
 2    64     1     3    110   211     0       0      144     1     1.8     1
 3    60     1     2    140   185     0       0      155     0     3       1
 4    56     1     3    120   193     0       0      162     0     1.9     1
 5    57     0     0    140   241     0       1      123     1     0.2     1
 6    59     1     3    170   288     0       0      159     0     0.2     1
 7    57     1     0    152   274     0       1       88     1     1.2     1
 8    57     1     0    130   131     0       1      115     1     1.2     1
 9    64     1     3    170   227     0       0      155     0     0.6     1
10    58     0     0    100   248     0       0      122     0     1       1
# i 232 more rows
# i 3 more variables: caa <dbl>, thall <dbl>, output <dbl>
```

This is the first 10 rows in the test set:

```
test
```

```
# A tibble: 61 x 14
     age   sex    cp trtbps  chol   fbs restecg thalachh  exng oldpeak   slp
   <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>   <dbl>    <dbl> <dbl>   <dbl> <dbl>
 1    37     1     2    130   250     0       1      187     0     3.5     0
 2    41     0     1    130   204     0       0      172     0     1.4     2
 3    48     0     2    130   275     0       1      139     0     0.2     2
 4    58     0     3    150   283     1       0      162     0     1       2
 5    43     1     0    150   247     0       1      171     0     1.5     2
 6    51     1     2    110   175     0       1      123     0     0.6     2
 7    54     1     2    150   232     0       0      165     0     1.6     2
 8    53     0     0    130   264     0       0      143     0     0.4     1
 9    44     1     2    140   235     0       0      180     0     0       2
10    53     0     2    128   216     0       0      115     0     0       2
# i 51 more rows
# i 3 more variables: caa <dbl>, thall <dbl>, output <dbl>
```

After separating the original data set into train and test sets, I used the glm() function to train the logistic regression model.

```r
logisticR_model <- glm(output ~ ., data=train, family = "binomial")
logisticR_model
```

```
Call:  glm(formula = output ~ ., family = "binomial", data = train)

Coefficients:
(Intercept)          age          sex           cp       trtbps         chol
   4.555347    -0.013283    -1.540661     0.769378    -0.017916    -0.003182
        fbs      restecg     thalachh         exng      oldpeak          slp
   0.132748     0.658516     0.019761    -1.077694    -0.678520     0.263729
        caa        thall
  -0.646435    -1.015510

Degrees of Freedom: 241 Total (i.e. Null);  228 Residual
Null Deviance:      333.5
Residual Deviance: 173.1    AIC: 201.1
```

This is the summary of the regression model that I have trained. In here we can see a lot of information about the model like estimation of the feature,residuals etc.

```r
summary(logisticR_model)
```

```
Call:
glm(formula = output ~ ., family = "binomial", data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.6127  -0.4311   0.1795   0.5835   2.4205

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  4.555347   2.794683   1.630  0.10310
age         -0.013283   0.025289  -0.525  0.59941
sex         -1.540661   0.509861  -3.022  0.00251 **
cp           0.769378   0.195309   3.939 8.17e-05 ***
trtbps      -0.017916   0.011003  -1.628  0.10348
chol        -0.003182   0.004177  -0.762  0.44617
fbs          0.132748   0.566248   0.234  0.81465
```

```
restecg      0.658516   0.389009   1.693  0.09049 .
thalachh     0.019761   0.011439   1.727  0.08408 .
exng        -1.077694   0.448609  -2.402  0.01629 *
oldpeak     -0.678520   0.240961  -2.816  0.00486 **
slp          0.263729   0.411859   0.640  0.52195
caa         -0.646435   0.201346  -3.211  0.00132 **
thall       -1.015510   0.330194  -3.075  0.00210 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 333.48  on 241  degrees of freedom
Residual deviance: 173.07  on 228  degrees of freedom
AIC: 201.07

Number of Fisher Scoring iterations: 6
```

Measuring Model Performance

In order to measure the performance of the trained model, we need to use the model with the test data and compare the results with the actual results.

For this, I first extracted the line to be predicted from the test data.

Then I use logistic regression model to predict the output in the test set.

```r
predicted_probs <- predict(logisticR_model, test[,-14],type = "response")
```

This is the top 14 data predicted by the regression model.

```r
head(predicted_probs)
```

```
        1         2         3         4         5         6
0.7079596 0.9397678 0.9823383 0.9748455 0.6193038 0.9260086
```

```r
predicted_classes <- ifelse(predicted_probs > 0.5, 1 , 0)
head(predicted_classes)
```

```
1 2 3 4 5 6
1 1 1 1 1 1
```

```r
TP <- sum(predicted_classes[which(test$output == 1)] == 1)
FP <- sum(predicted_classes[which(test$output == 1)] == 0)
TN <- sum(predicted_classes[which(test$output == 0)] == 0)
FN <- sum(predicted_classes[which(test$output == 0)] == 1)

specificity <- TN / (TN + FP)
recall <- TP / (TP + FN)
accuracy <- (TN + TP) / (TP + FP + TN + FN)
precision <- TP / (TP + FP)

recall
```

```
[1] 0.7948718
```

```r
specificity
```

```
[1] 0.9090909
```

```r
precision
```

```
[1] 0.9393939
```

```r
accuracy
```

```
[1] 0.8360656
```

Performance values tell us some metrics about the model. The model classifies the observations with 0.83 accuracy. Its precision is 0.93 means that the model classifies only 93% of the positive class, (lower risk of the heart attack)

```r
table(train$output) / dim(train)[1]
```

```
        0         1 
0.4545455 0.5454545 
```

As shown in the above result, values are balanced.

```r
confusionMatrix(table(ifelse(test$output == "1" , "1" , "0"),
                      predicted_classes),
                positive = "1")
```

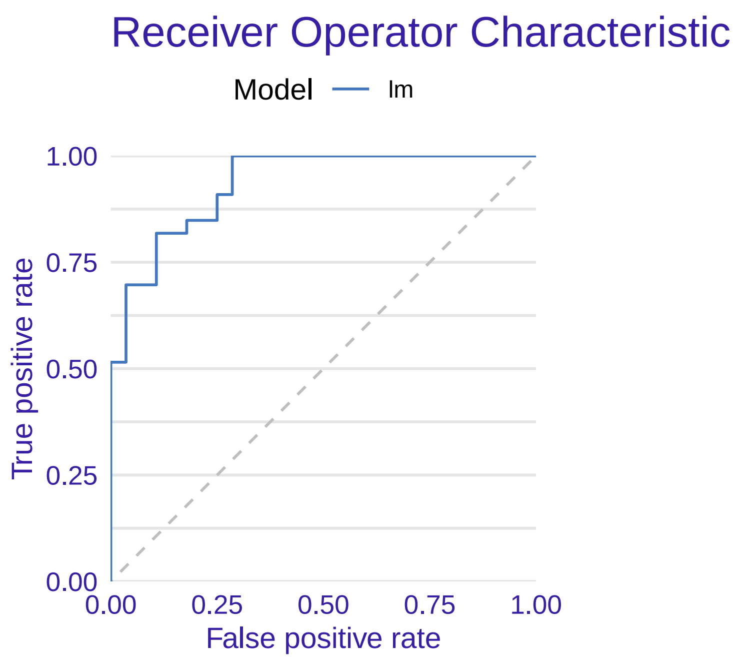
```
Confusion Matrix and Statistics

   predicted_classes
     0  1
  0 20  8
  1  2 31
                                          
               Accuracy : 0.8361          
                 95% CI : (0.7191, 0.9185)
    No Information Rate : 0.6393          
    P-Value [Acc > NIR] : 0.000614        
                                          
                  Kappa : 0.6645          
                                          
 Mcnemar's Test P-Value : 0.113846        
                                          
            Sensitivity : 0.7949          
            Specificity : 0.9091          
         Pos Pred Value : 0.9394          
         Neg Pred Value : 0.7143          
             Prevalence : 0.6393          
         Detection Rate : 0.5082          
   Detection Prevalence : 0.5410          
      Balanced Accuracy : 0.8520          
                                          
       'Positive' Class : 1               
```

ROC Curve

```
explain_logisticR <- DALEX::explain(model = logisticR_model,
                          data = test [, -14],
                          y    = test$output == "1",
                          type = "classification",
                          verbose = FALSE)
```

```
performance_logiscticR <- model_performance(explain_logisticR)
plot(performance_logiscticR, geom= "roc")
```




```
performance_logiscticR
```

```
Measures for:  classification
recall     : 0.9393939
precision  : 0.7948718
f1         : 0.8611111
accuracy   : 0.8360656
auc        : 0.9339827
```

```
Residuals:
         0%         10%         20%         30%         40%         50% 
-0.92500426 -0.66440247 -0.23803316 -0.05420990 -0.00710291  0.01057154 
        60%         70%         80%         90%        100% 
 0.02908870  0.05916045  0.19665257  0.30022165  0.59491469 
```

The area under the curve is equal to 0.93. It is a good value because the max value it can have is 1. The actual value is very close to the max value so our model's performance is nearly perfect.