Introduction to Logistic Regression (Part 2)

Heaven Klair

5/6/2022

= ## Importing the dataset

```
url <- "http://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/processed.cleveland.data"
dataset = read.csv(url, header=FALSE)
head(dataset)
```

```
##   V1 V2 V3  V4  V5 V6 V7  V8 V9 V10 V11 V12 V13 V14
## 1 63  1  1 145 233  1  2 150  0 2.3   3 0.0 6.0   0
## 2 67  1  4 160 286  0  2 108  1 1.5   2 3.0 3.0   2
## 3 67  1  4 120 229  0  2 129  1 2.6   2 2.0 7.0   1
## 4 37  1  3 130 250  0  0 187  0 3.5   3 0.0 3.0   0
## 5 41  0  2 130 204  0  2 172  0 1.4   1 0.0 3.0   0
## 6 56  1  2 120 236  0  0 178  0 0.8   1 0.0 3.0   0
```

After importing the data, we see that none of the columns are labeled. So we can name the columns looking at the UCI website.

```
colnames(dataset) <- c("age", "sex", "cp", "trestbps", "chol", "fbs",
                       "restecg", "thalach", "exang", "oldpeak", "slope", "ca",
                       "thal", "hd")
head(dataset)
```

```
##   age sex cp trestbps chol fbs restecg thalach exang oldpeak slope  ca thal hd
## 1  63   1  1      145  233   1       2     150     0     2.3     3 0.0  6.0  0
## 2  67   1  4      160  286   0       2     108     1     1.5     2 3.0  3.0  2
## 3  67   1  4      120  229   0       2     129     1     2.6     2 2.0  7.0  1
## 4  37   1  3      130  250   0       0     187     0     3.5     3 0.0  3.0  0
## 5  41   0  2      130  204   0       2     172     0     1.4     1 0.0  3.0  0
## 6  56   1  2      120  236   0       0     178     0     0.8     1 0.0  3.0  0
```

```
str(dataset)
```

```
## 'data.frame':    303 obs. of  14 variables:
##  $ age     : num  63 67 67 37 41 56 62 57 63 53 ...
##  $ sex     : num  1 1 1 1 0 1 0 0 1 1 ...
##  $ cp      : num  1 4 4 3 2 2 4 4 4 4 ...
##  $ trestbps: num  145 160 120 130 130 120 140 120 130 140 ...
##  $ chol    : num  233 286 229 250 204 236 268 354 254 203 ...
##  $ fbs     : num  1 0 0 0 0 0 0 0 0 1 ...
##  $ restecg : num  2 2 2 0 2 0 2 0 2 2 ...
##  $ thalach : num  150 108 129 187 172 178 160 163 147 155 ...
##  $ exang   : num  0 1 1 0 0 0 0 1 0 1 ...
##  $ oldpeak : num  2.3 1.5 2.6 3.5 1.4 0.8 3.6 0.6 1.4 3.1 ...
##  $ slope   : num  3 2 2 3 1 1 3 1 2 3 ...
##  $ ca      : chr  "0.0" "3.0" "2.0" "0.0" ...
```

```
##  $ thal    : chr  "6.0" "3.0" "7.0" "3.0" ...
##  $ hd      : int  0 2 1 0 0 0 3 0 2 1 ...
```

Now, we have column names in the dataset. But when we run str() function, we see that some of the columns are messed up. First, sex is a number in the dataset, but it is supposes to be a factor, where 0 represents "female" and 1 represents "male". cp (Chest Pain) is also supposed to be a factor where levels 1-3 represents different types of pain and 4 represents no chest pain. So let' clean the data before we start to use it.

```
dataset[dataset$sex == 0,]$sex <- "F"
dataset[dataset$sex == 1,]$sex <- "M"

# Now we convert the column into a factor
dataset$sex <- as.factor(dataset$sex)

# Convert a bunch of other columns into factors
dataset$cp <- as.factor(dataset$cp)
dataset$fbs <- as.factor(dataset$fbs)
dataset$restecg <- as.factor(dataset$restecg)
dataset$exang <- as.factor(dataset$exang)
dataset$slope <- as.factor(dataset$slope)
```

Since the ca column is a column of strings (chars), as seen when we ran str(dataset). R things that is a column of char, but instead it has values of integers. We correct that assumption by telling R that it is a column of integers.

```
dataset$ca <- as.integer(dataset$ca)
```

```
## Warning: NAs introduced by coercion
```

```
dataset$ca <- as.factor(dataset$ca)

# "Thal" column needs the similar correction
dataset$thal <- as.integer(dataset$thal)
```

```
## Warning: NAs introduced by coercion
```

```
dataset$thal <- as.factor(dataset$thal)
```

The last thing that we need to do is make *hd* (Heart Disease), a factor that is easy on the eyes. Here, we are going to use a fancy trick with ifelse() to convert the 0's to "Healthy" and 1's to "Unhealthy".

```
dataset$hd <- ifelse(test = dataset$hd == 0, yes = "Healthy", no="Unhealthy")
dataset$hd <- as.factor(dataset$hd)
```

Let us check if we have made all the neccessary changes to the data

```
str(dataset)
```

```
## 'data.frame':    303 obs. of  14 variables:
##  $ age     : num  63 67 67 37 41 56 62 57 63 53 ...
##  $ sex     : Factor w/ 2 levels "F","M": 2 2 2 2 1 2 1 1 2 2 ...
##  $ cp      : Factor w/ 4 levels "1","2","3","4": 1 4 4 3 2 2 4 4 4 4 ...
##  $ trestbps: num  145 160 120 130 130 120 140 120 130 140 ...
##  $ chol    : num  233 286 229 250 204 236 268 354 254 203 ...
##  $ fbs     : Factor w/ 2 levels "0","1": 2 1 1 1 1 1 1 1 1 2 ...
##  $ restecg : Factor w/ 3 levels "0","1","2": 3 3 3 1 3 1 3 1 3 3 ...
##  $ thalach : num  150 108 129 187 172 178 160 163 147 155 ...
##  $ exang   : Factor w/ 2 levels "0","1": 1 2 2 1 1 1 1 2 1 2 ...
##  $ oldpeak : num  2.3 1.5 2.6 3.5 1.4 0.8 3.6 0.6 1.4 3.1 ...
```

```
##  $ slope   : Factor w/ 3 levels "1","2","3": 3 2 2 3 1 1 3 1 2 3 ...
##  $ ca      : Factor w/ 4 levels "0","1","2","3": 1 4 3 1 1 1 3 1 2 1 ...
##  $ thal    : Factor w/ 3 levels "3","6","7": 2 1 3 1 1 1 1 1 3 3 ...
##  $ hd      : Factor w/ 2 levels "Healthy","Unhealthy": 1 2 2 1 1 1 2 1 2 2 ...
```

It looks like we have made the appropriate changes and are ready to proceed further.

Now we see how many samples (rows of data) have NA values. Later we will decide if we can just toss these samples out, or if we should impute values for the NAs.

```
nrow(dataset[is.na(dataset$ca) | is.na(dataset$thal), ])
```

```
## [1] 6
```

We get that there are 6 samples that have NAs in them. Let us view the samples with NAs by selecting those rows from the dataframe

```
dataset[is.na(dataset$ca) | is.na(dataset$thal), ]
```

```
##     age sex cp trestbps chol fbs restecg thalach exang oldpeak slope   ca thal
## 88   53   F  3      128  216   0       2     115     0     0.0     1    0 <NA>
## 167  52   M  3      138  223   0       0     169     0     0.0     1 <NA>    3
## 193  43   M  4      132  247   1       2     143     1     0.1     2 <NA>    7
## 267  52   M  4      128  204   1       0     156     1     1.0     2    0 <NA>
## 288  58   M  2      125  220   0       0     144     0     0.4     2 <NA>    7
## 303  38   M  3      138  175   0       0     173     0     0.0     1 <NA>    3
##            hd
## 88    Healthy
## 167   Healthy
## 193 Unhealthy
## 267 Unhealthy
## 288   Healthy
## 303   Healthy
```

Since there are only 6 samples, we will go ahead and remove them from the dataset and proceed further.

```
dataset <- dataset[!(is.na(dataset$ca) | is.na(dataset$thal) ), ]
nrow(dataset)
```

```
## [1] 297
```

Now, let us make sure that healthy and diseased samples come from each gender (male and female). If only male samples have heart diseases, we should probably remove all females from the model. We can do this with the xtabs() function. Since we want a table with heart diseases and sex, so we pass those two columns into the function.

```
xtabs(~ hd + sex, data = dataset)
```

```
##            sex
## hd            F   M
##   Healthy    71  89
##   Unhealthy  25 112
```

Healthy and Unhealthy patients are both represented by a lot of female and male samples.

Now, let us verify that all 4 levels of Chest pain (cp) were reported by a bunch of patients.

```
xtabs(~ hd + cp, data = dataset)
```

```
##            cp
## hd            1   2   3   4
```

```
##   Healthy    16  40  65  39
##   Unhealthy   7   9  18 103
```

We do this for all of the boolean and categorical variables that we are using to predict heart diseases.

```
xtabs(~ hd + fbs, data = dataset)
```

```
##            fbs
## hd            0   1
##   Healthy   137  23
##   Unhealthy 117  20
```

```
xtabs(~ hd + restecg, data = dataset)
```

```
##            restecg
## hd           0  1  2
##   Healthy   92  1 67
##   Unhealthy 55  3 79
```

We find here something that could cause trouble for the "restecg". Only 4 patients represents level 1. This could, potentially, get in the way of finding the best fitting line. However, for now we will just leave it in and see what happens.

```
xtabs(~ hd + exang, data = dataset)
```

```
##            exang
## hd            0   1
##   Healthy   137  23
##   Unhealthy  63  74
```

```
xtabs(~ hd + slope, data = dataset)
```

```
##            slope
## hd            1   2   3
##   Healthy   103  48   9
##   Unhealthy  36  89  12
```

```
xtabs(~ hd + ca, data = dataset)
```

```
##            ca
## hd            0   1   2   3
##   Healthy   129  21   7   3
##   Unhealthy  45  44  31  17
```

```
xtabs(~ hd + thal, data = dataset)
```

```
##            thal
## hd            3   6   7
##   Healthy   127   6  27
##   Unhealthy  37  12  88
```

Now, we have looked at all of them, we are ready to do Logistic Regression.

Logistic Regression with one Indepedent Variable

We will try to predict heart disease using only the gender of each patient.

```
classifier <- glm(hd ~ sex, data = dataset, family = binomial )
summary(classifier)
```

```
##
```

```
## Call:
## glm(formula = hd ~ sex, family = binomial, data = dataset)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.2765  -1.2765  -0.7768   1.0815   1.6404
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -1.0438     0.2326  -4.488 7.18e-06 ***
## sexM          1.2737     0.2725   4.674 2.95e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 409.95  on 296  degrees of freedom
## Residual deviance: 386.12  on 295  degrees of freedom
## AIC: 390.12
##
## Number of Fisher Scoring iterations: 4
```

The coefficients respond to the following:

$$\text{Heart Disease} = -1.0438 + 1.2737 \times \text{the patient is male}$$

We know that patient is male equals to being 1. And if patient is female, then the value of the patient = 0. So in the above equation, let us predict heart disease for a female, we get the following equation:

$$\text{Heart Disease} = -1.0438 + 1.2737 \times 0$$

The heart disease comes out to be log(odds) $= -1.0438$.

For a male patient, the equation becomes

$$\text{Heart Disease} = -1.0438 + 1.2737 \times 1$$

$$\text{Heart Disease} = -1.0438 + 1.2737$$

Since the first term above is the log(odds) of a female having a heart disease, the second term indicates the increase in the log(odds) that a male has of having heart disease. In other words, the second term is the log(odds ratio) of the odds that a male will have heart disease over the odds that a female will have heart disease.

Both the p-values are well below 0.05, and thus, the log(odds) and the log(odds ratio) are both statistically significant.

We see an extra line in the summary of the logistic function which says: "Dispersion parameter for binomial family taken to be 1". When we do "normal" linear regression we estimate both the mean and the variance from the data. In contrast, with logistic regression, we estimate the mean of the data, and the variance is derived from the mean. Since we are not estimating the variance from the data (and, instead just deriving it from the mean), it is possible that the variance is underestimated. If so, we can adjust the dispersion parameter in the summary() command.

Then we have "Null deviance" and "Residual deviance". These can be used to compare models, compute R^2 and an overall p-value.

Then we have "AIC" (Akaike Information Criterion), which, in this context, is just the Residual Deviance adjusted for the number of parameters in the model. The AIC can be used to compare one model to the other.

Lastly, we have "Number of Fisher Scoring iterations", which just tells us how quickly the glm() function converged on the maximum likelihood estimates for the coefficients.

Logistic Regression with many independent variables

We will use all the variables to predict the heart disease.

```
classifier <- glm(hd ~., data = dataset, family = binomial)
summary(classifier)
```

```
##
## Call:
## glm(formula = hd ~ ., family = binomial, data = dataset)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.0490  -0.4847  -0.1213   0.3039   2.9086
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -6.253978   2.960399  -2.113 0.034640 *
## age         -0.023508   0.025122  -0.936 0.349402
## sexM         1.670152   0.552486   3.023 0.002503 **
## cp2          1.448396   0.809136   1.790 0.073446 .
## cp3          0.393353   0.700338   0.562 0.574347
## cp4          2.373287   0.709094   3.347 0.000817 ***
## trestbps     0.027720   0.011748   2.359 0.018300 *
## chol         0.004445   0.004091   1.087 0.277253
## fbs1        -0.574079   0.592539  -0.969 0.332622
## restecg1     1.000887   2.638393   0.379 0.704424
## restecg2     0.486408   0.396327   1.227 0.219713
## thalach     -0.019695   0.011717  -1.681 0.092781 .
## exang1       0.653306   0.447445   1.460 0.144267
## oldpeak      0.390679   0.239173   1.633 0.102373
## slope2       1.302289   0.486197   2.679 0.007395 **
## slope3       0.606760   0.939324   0.646 0.518309
## ca1          2.237444   0.514770   4.346 1.38e-05 ***
## ca2          3.271852   0.785123   4.167 3.08e-05 ***
## ca3          2.188715   0.928644   2.357 0.018428 *
## thal6       -0.168439   0.810310  -0.208 0.835331
## thal7        1.433319   0.440567   3.253 0.001141 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 409.95  on 296  degrees of freedom
## Residual deviance: 183.10  on 276  degrees of freedom
## AIC: 225.1
##
## Number of Fisher Scoring iterations: 6
```

1. We see that age is not a useful predictor because it has a large p-value. However, the median age in our dataset was 56, so most of the folks were pretty old and that explains why it was not very useful.
2. Gender is still a good predictor tho.
3. The Residual Deviance and the AIC are both much smaller for this fancy model than they were for the simple model, when we only used gender to predict heart disease.

McFadden's Pseudo R^2

If we want to calculate McFadden's Pseudo R^2 we can pull the log-likelihood of the null model out of the logistic variable by getting the value for the null deviance and dividing by -2 and we can pull the log-likelihood for the fancy model out of the logistic variable by getting the value of residual deviance and dividing by -2.

```
ll.null <- classifier$null.deviance/(-2)

ll.proposed <- classifier$deviance/(-2)

(ll.null - ll.proposed)/ll.null
```

```
## [1] 0.5533531
```

Thus, we get a Pseudo $R^2 = 0.55$. This can be interpreted as the overall effect size.

And we can use those same log-likelihoods to calculate a p-value for that R^2 using a Chi-square distribution.

```
1 - pchisq(2*(ll.proposed - ll.null), df=(length(classifier$coefficients)-1))
```

```
## [1] 0
```

In this case, the p-value is tiny, so the R^2 value isnt due to dumb luck.

Graphing the predicted probabilities

Lastly, we can draw a graph that shows the predicted probabilities that each patient has a heart disease along with their actual heart status. To draw the graph, we start by creating a new dataframe that contains the probabilities of having heart disease along with the actual heart disease status.

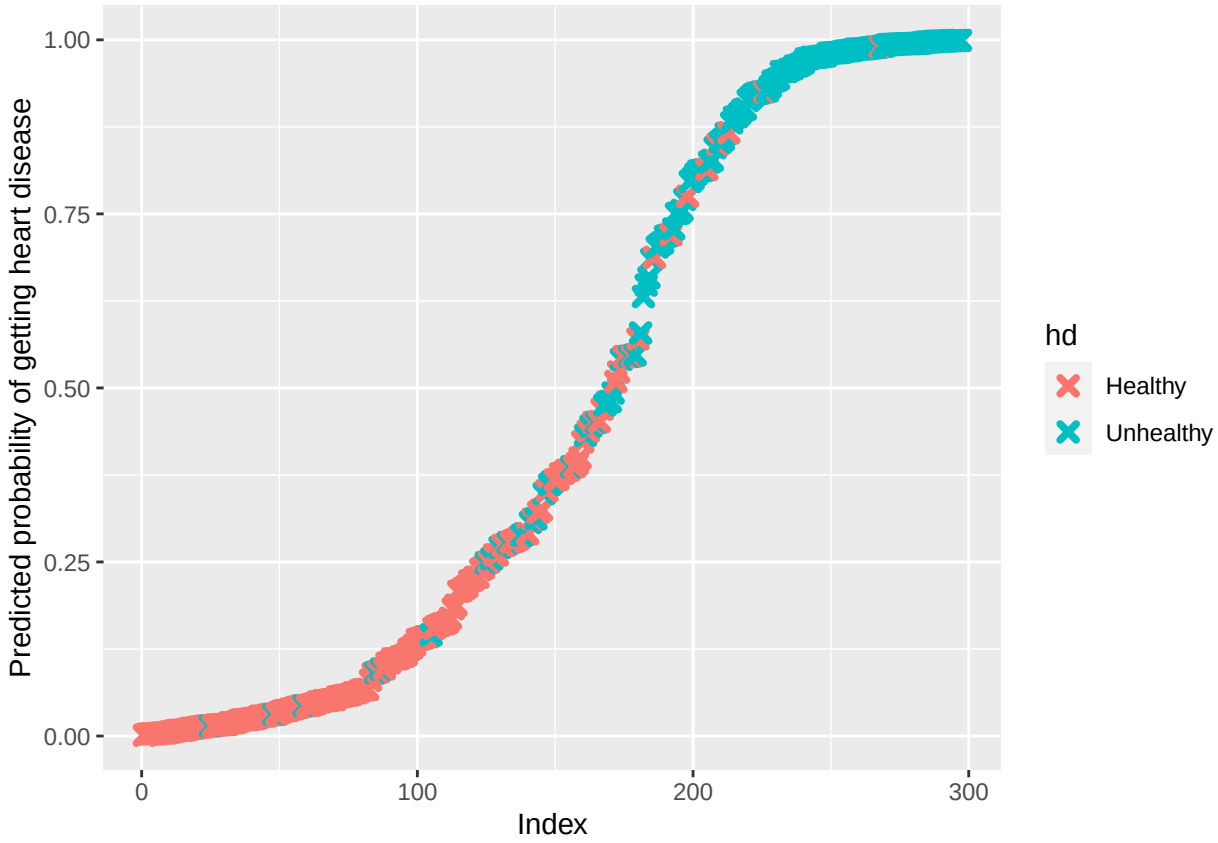
Then we sort the data.frame from low probability to high probability.

We add a new column to the data.frame that has the rank of each sample, from low probability to high probility.

```
predicted.dataset <- data.frame(
  probability.of.hd=classifier$fitted.values,
  hd = dataset$hd
)

predicted.dataset <- predicted.dataset[
  order(predicted.dataset$probability.of.hd, decreasing = FALSE),]

predicted.dataset$rank <- 1:nrow(predicted.dataset)
```

```
library(ggplot2)

#install.packages('cowplot')
library(cowplot) # load cowplot library so that ggplot has nice looking defaults

ggplot(data = predicted.dataset, aes(x=rank, y=probability.of.hd) ) +
  geom_point(aes(color=hd), alpha=1, shape=4, stroke=2)+
```

```
  xlab("Index")+
  ylab("Predicted probability of getting heart disease")
```



Most of the patients with heart disease (the ones in turquoise) are predicted to have high probability of having heart disease most of the patients without heart disease (the ones in salmon) are predicted to have a low probability of having heart disease. Thus, our logistic regression has done a pretty good job.

However, we could use cross-validation to get a better idea of how well it might perform with new data.