Linear Regression

Sebastian Heucke

9/30/2019

Contents

Logistic Regression

This tutorial decribes how to do Logistic Regression its a markdown version of the https://www.youtube.com/watch?v=C4N3_XJJ-jU&list=PLblh5JKOoLUJJpBNfk8_YadPwDTO2SCbx&index=8&t=0s video.

We start by importing a dataset and cleaning it up, then we perform logistic regression on a very simple model, followed by a fancy model. Lastly we draw a graph of the predicted probabilities that came from the Logistic Regression.

For the example, we are using a real dataset form the UCI machine learning repository: http://archive.ics.uci.edu/ml/index.php. Specifically, we want the Heart Disease Dataset: http://archive.ics.uci.edu/ml/datasets/Heart+Disease.

```r
# variable url with location in internet
url <- "http://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/processed.cleveland.data

# download data
data <- read.csv(url, header=FALSE)

# look at the first 6 rows
head(data)
```

```
##   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
```

Unfortunately, none of the columns are labeled, so we name the columns after the names that were listed on the UCI website.

```r
colnames(data) <- c(
  "age",
  "sex",# 0 = female, 1 = male
  "cp", # chest pain
  # 1 = typical angina,
  # 2 = atypical angina,
  # 3 = non-anginal pain,
  # 4 = asymptomatic
  "trestbps", # resting blood pressure (in mm Hg)
  "chol", # serum cholestoral in mg/dl
  "fbs",  # fasting blood sugar if less than 120 mg/dl, 1 = TRUE, 0 = FALSE
  "restecg", # resting electrocardiographic results
  # 1 = normal
  # 2 = having ST-T wave abnormality
  # 3 = showing probable or definite left ventricular hypertrophy
  "thalach", # maximum heart rate achieved
  "exang",   # exercise induced angina, 1 = yes, 0 = no
  "oldpeak", # ST depression induced by exercise relative to rest
  "slope", # the slope of the peak exercise ST segment
  # 1 = upsloping
  # 2 = flat
  # 3 = downsloping
  "ca", # number of major vessels (0-3) colored by fluoroscopy
  "thal", # this is short of thalium heart scan
  # 3 = normal (no cold spots)
  # 6 = fixed defect (cold spots during rest and exercise)
  # 7 = reversible defect (when cold spots only appear during exercise)
  "hd" # (the predicted attribute) - diagnosis of heart disease
  # 0 if less than or equal to 50% diameter narrowing
  # 1 if greater than 50% diameter narrowing
)

head(data) # now we have data and column names
```

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

This looks better. However the **str()** function, which describes the structure of the data, tells us that some of the columns are messed up.

```r
str(data)
```

```
## '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      : Factor w/ 5 levels "?","0.0","1.0",..: 2 5 4 2 2 2 4 2 3 2 ...
##  $ thal    : Factor w/ 4 levels "?","3.0","6.0",..: 3 2 4 2 2 2 2 2 4 4 ...
##  $ hd      : int  0 2 1 0 0 0 3 0 2 1 ...
```

Right now, **sex** is a number, but it is supposed to be a factor, where 0 represents "female" and 1 represents "male". **cp** (aka **c**hest **p**ain) is also supposed to be a factor, where levels 1-3 represent different types of pain and 4 represents no chest pain. **ca** and **thal** are correctly called factors, but one of the levels is **"?"** when we need it to be **NA**. So we have to clean up the data.

```r
data[data == "?"] <- NA
data[data$sex == 0,]$sex <- "F"
data[data$sex == 1,]$sex <- "M"
data$sex <- as.factor(data$sex)

data$cp <- as.factor(data$cp)
data$fbs <- as.factor(data$fbs)
data$restecg <- as.factor(data$restecg)
data$exang <- as.factor(data$exang)
data$slope <- as.factor(data$slope)

data$ca <- as.integer(data$ca) # since this column had "?"s in it
# R thinks that the levels for the factor are strings, but
# we know they are integers, so first convert the strings to integiers...
data$ca <- as.factor(data$ca)  # ...then convert the integers to factor levels

data$thal <- as.integer(data$thal) # "thal" also had "?"s in it.
data$thal <- as.factor(data$thal)

## This next line replaces 0 and 1 with "Healthy" and "Unhealthy"
data$hd <- ifelse(test=data$hd == 0, yes="Healthy", no="Unhealthy")
data$hd <- as.factor(data$hd) # Now convert to a factor

str(data) ## this shows that the correct columns are factors
```

```
## '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 "2","3","4","5": 1 4 3 1 1 1 3 1 2 1 ...
##  $ thal    : Factor w/ 3 levels "2","3","4": 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 ...
```

Now we are able to see how many samples (rows of data) have **NA** values.

```
# Now determine how many rows have "NA" (aka "Missing data"). If it's just
# a few, we can remove them from the dataset, otherwise we should consider
# imputing the values with a Random Forest or some other imputation method.
nrow(data[is.na(data$ca) | is.na(data$thal),])
```

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

6 Samples (rows of data) have **NAs** in them.

We can view the samples with **NAs** by selecting those rows from the data.frame.

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

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

```
## so 6 of the 303 rows of data have missing values. This isn't a large
## percentage (2%), so we can just remove them from the dataset
## NOTE: This is different from when we did machine learning with
## Random Forests. When we did that, we imputed values.
nrow(data)
```

```
## [1] 303
```

If we wanted to, we could impute values for the **NAs** using a Random Forest or some other method. However, for this example, we'll just remove these samples.

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

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

After removing those samples (rows), there are 297 samples remaining.

Now we need to make sure that healthy and diseased samples come from each gender (female and male). If only male have heart disease, we should probably remove all females from the model. Do the same thing with the remaining variables to make sure that they are all represented by a number of patients.

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

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

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

```
##            cp
## hd            1   2   3   4
##   Healthy    16  40  65  39
##   Unhealthy   7   9  18 103
```

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

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

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

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

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

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

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

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

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

```
##            ca
## hd            2   3   4   5
##   Healthy   129  21   7   3
##   Unhealthy  45  44  31  17
```

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

```
##            thal
## hd            2   3   4
##   Healthy   127   6  27
##   Unhealthy  37  12  88
```

Logistic Regression

Start with a simple model. We will try to predict heart disease using only the gender of each patient.

The function for **G**eneralized **L**inear **M**odels is called **glm()**. First we use formula syntax to specify that we want to use sex to predict heart disease (**hd**). Then specify the data and lastly specify that we want the **binomial** family of generalize linear models. This makes the **glm()** function do Logistic Regression, as apposed to some other type of generalized linear model. The output from the **glm()** function is stored in the variable called "**logistic**". We then use the **summary() function to get details about the logistic regression.

```
logistic <- glm(hd ~ sex, data=data, family="binomial")
summary(logistic)
```

```
##
## Call:
## glm(formula = hd ~ sex, family = "binomial", data = data)
##
## 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 first line has the original call to the **glm()** function. Then it gives you a summary of the deviance residuals. They should center on zero and look symetrical. Then we have the coefficients. They correspond to the following model:

$HeartDisease = -1.0438 + 1.2737 \times ThePatientIsMale$

The variable (the patient is male) is equal to 0 when the patient is female and 1 when the patient is male. Thus, if we are predicting heart disease for a female patient, we get the following equqtion:

$HeartDisease = -1.0438 + 1.2737 \times 0$

```
## (Intercept)  -1.0438     0.2326  -4.488 7.18e-06 ***
##   sexM        1.2737     0.2725   4.674 2.95e-06 ***

## Let's start by going through the first coefficient...
## (Intercept)  -1.0438     0.2326  -4.488 7.18e-06 ***
##
## The intercept is the log(odds) a female will be unhealthy. This is because
## female is the first factor in "sex" (the factors are ordered,
## alphabetically by default,"female", "male")
female.log.odds <- log(25 / 71)
female.log.odds
```

```
## [1] -1.043804
```

Thus, the log (odds) that a female has heart disease is -1.0438. I we are predicting heart disease for a male patient, we get the following equation:

$HeartDisease = -1.0438 + 1.2737 \times 1$

```
## Now let's look at the second coefficient...
##   sexM        1.2737     0.2725   4.674 2.95e-06 ***
##
## sexM is the log(odds ratio) that tells us that if a sample has sex=M, the
## odds of being unhealthy are, on a log scale, 1.27 times greater than if
## a sample has sex=F.
male.log.odds.ratio <- log((112 / 89) / (25/71))
male.log.odds.ratio
```

```
## [1] 1.273667
```

Since this first term is log (odds) of a female having 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.

The second part of the output shows the Wald's test was computed for both coefficients and the p-valus. Both p-values are well below 0.05, and thus, the log (odds) and the log (odds ratios) are both statistically significant.

But, remember, a small p-value alone isn't interesting, we also want large effect sizes, and that's what the log (odds) and log (odds ratio) tells us.

The next line in the output shows the default dispersion parameter used for this logistic regression. 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, it is possible that the variance is underestimated. If so, you can adkust the dispersion parameter in the **summary()** command.

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

Next we have the AIC, the **A**kaike **I**nformation **C**riterion, 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 another.

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

```
library(ggplot2)
## Now calculate the overall "Pseudo R-squared" and its p-value

## NOTE: Since we are doing logistic regression...
## Null deviance = 2*(0 - LogLikelihood(null model))
##               = -2*LogLikihood(null model)
## Residual deviane = 2*(0 - LogLikelihood(proposed model))
##                  = -2*LogLikelihood(proposed model)
ll.null <- logistic$null.deviance/-2
ll.proposed <- logistic$deviance/-2

## McFadden's Pseudo R^2 = [ LL(Null) - LL(Proposed) ] / LL(Null)
(ll.null - ll.proposed) / ll.null
```

```
## [1] 0.05812569
```

```
## chi-square value = 2*(LL(Proposed) - LL(Null))
## p-value = 1 - pchisq(chi-square value, df = 2-1)
1 - pchisq(2*(ll.proposed - ll.null), df=1)
```

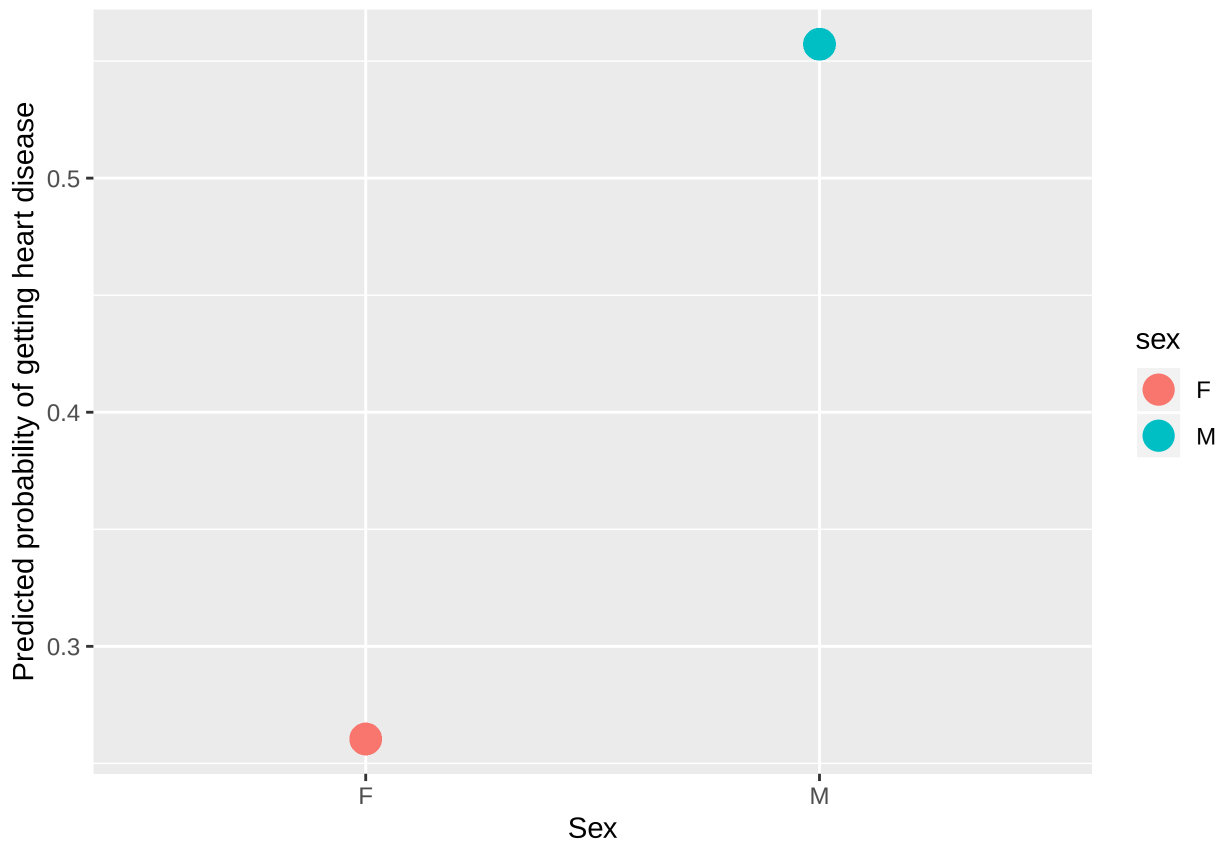
```
## [1] 1.053157e-06
```

```
1 - pchisq((logistic$null.deviance - logistic$deviance), df=1)
```

```
## [1] 1.053157e-06
```

```
## Lastly, let's  see what this logistic regression predicts, given
## that a patient is either female or male (and no other data about them).
predicted.data <- data.frame(
  probability.of.hd=logistic$fitted.values,
  sex=data$sex)

## We can plot the data...
ggplot(data=predicted.data, aes(x=sex, y=probability.of.hd)) +
  geom_point(aes(color=sex), size=5) +
  xlab("Sex") +
  ylab("Predicted probability of getting heart disease")
```



```
## Since there are only two probabilities (one for females and one for males),
## we can use a table to summarize the predicted probabilities.
xtabs(~ probability.of.hd + sex, data=predicted.data)
```

```
##                    sex
## probability.of.hd     F   M
##   0.260416666667241  96   0
##   0.55721393034826    0 201
```

A more fancy model

Now that we've done a simple logistic regression using just one of the variables (**sex**) to predict heart disease, we can create a fancy model that uses all of the variables to predict heart disease.

```
logistic <- glm(hd ~ ., data=data, family="binomial")
summary(logistic)
```

```
##
## Call:
## glm(formula = hd ~ ., family = "binomial", data = data)
##
## 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
## ca3          2.237444   0.514770   4.346 1.38e-05 ***
## ca4          3.271852   0.785123   4.167 3.08e-05 ***
## ca5          2.188715   0.928644   2.357 0.018428 *
## thal3       -0.168439   0.810310  -0.208 0.835331
## thal4        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
```

This formula syntax, **hd~.**, means that we want to model heart disease (**hd**) using all of the remaining variables in our data.frame called "data". We can then see what our model looks like with the **summary()** function.

We see that **age** isn't a useful predictor because it as 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 wasn't very useful. Gender us still a good output.

If we look at the bottom of the output we see that 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.

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 for the **residual deviance** and dividing by -2.

Then we just do the math and we end up with a Pseudo $R^2 = 0.55$. This can be interpreted as the overall effect size. Then we can use those same log-likelihoods to calculate a p-value for that R^2 using a Chi-square distribution. In this case the p-value is tiny, so the R^2 value isn't due to luck.

Lastly, we can draw a graph that shows the predicted probabilities that each patient has heart disease along with their actual heart disease status. Most of the patients with heart disease (the ones in *turquoise*), are predicted to have a **high** probability of having heart disease and most of the patients without heart disease (the ones in *salmon*), are predicted to have a **low** probability of having heart disease. Thus the logistic regression did a good job. However we could use cross-validation to get a better idea of how well it might perform with new data.

To draw the graph, we start by creating a new data.frame that contains the probabilities of having heart desease along with actual heart disease status. Then we sort the data.frame from low probabilities to high probabilities. Next we add a new column to the data.frame that has the **rank** of each sample, from low probability to high probability. Then we load the ggplot2 library so we can draw a graph. We load the **cowplot** library so that ggplot has nice looking defaults. We call **ggplot()** and use **geom_point()** to draw the data and lastly, we call **ggsave()** to save the graph as PDF file.

```
## Now calculate the overall "Pseudo R-squared" and its p-value
ll.null <- logistic$null.deviance/-2
ll.proposed <- logistic$deviance/-2

## McFadden's Pseudo R^2 = [ LL(Null) - LL(Proposed) ] / LL(Null)
(ll.null - ll.proposed) / ll.null
```

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

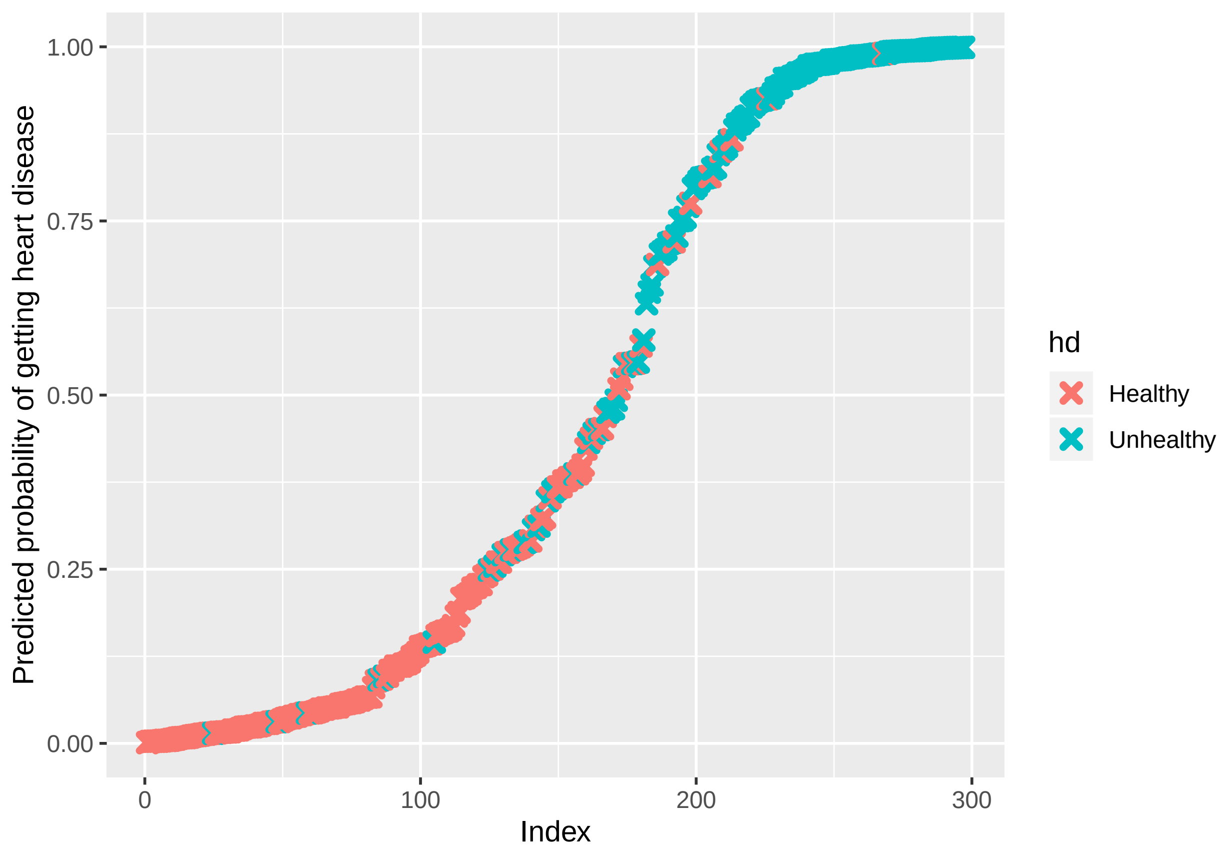
```
## The p-value for the R^2
1 - pchisq(2*(ll.proposed - ll.null), df=(length(logistic$coefficients)-1))
```

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

```
## now we can plot the data
predicted.data <- data.frame(
  probability.of.hd=logistic$fitted.values,
  hd=data$hd)

predicted.data <- predicted.data[
  order(predicted.data$probability.of.hd, decreasing=FALSE),]
predicted.data$rank <- 1:nrow(predicted.data)

## Lastly, we can plot the predicted probabilities for each sample having
## heart disease and color by whether or not they actually had heart disease
ggplot(data=predicted.data, 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")
```



```
ggsave("heart_disease_probabilities.pdf")
```

```
## Saving 6.5 x 4.5 in image
```

Session information

```
## R version 3.6.1 (2019-07-05)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 18.04.3 LTS
##
## Matrix products: default
## BLAS:   /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.7.1
## LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.7.1
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
```

```
## [1] ggplot2_3.2.0  knitr_1.23     devtools_2.1.0 usethis_1.5.1
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.1        pillar_1.4.2      compiler_3.6.1
##  [4] prettyunits_1.0.2 remotes_2.1.0     tools_3.6.1
##  [7] testthat_2.1.1    digest_0.6.20     pkgbuild_1.0.5
## [10] pkgload_1.0.2     tibble_2.1.3      evaluate_0.14
## [13] memoise_1.1.0     gtable_0.3.0      pkgconfig_2.0.2
## [16] rlang_0.4.0       cli_1.1.0         yaml_2.2.0
## [19] xfun_0.8          dplyr_0.8.3       withr_2.1.2
## [22] stringr_1.4.0     desc_1.2.0        fs_1.3.1
## [25] tidyselect_0.2.5  rprojroot_1.3-2   grid_3.6.1
## [28] glue_1.3.1        R6_2.4.0          processx_3.4.1
## [31] rmarkdown_1.14    sessioninfo_1.1.1 purrr_0.3.2
## [34] callr_3.3.1       magrittr_1.5      backports_1.1.4
## [37] scales_1.0.0      ps_1.3.0          htmltools_0.3.6
## [40] assertthat_0.2.1  colorspace_1.4-1  labeling_0.3
## [43] stringi_1.4.3     lazyeval_0.2.2    munsell_0.5.0
## [46] crayon_1.3.4
```

The document was processed on 2019-10-01.