Training The Logistic Regression Model on The "Pimaındiansdiabetes" Data And Interpreting The Performance of The Model

Merve Topdemir

18 03 2021

CONTENTS

I'll be using the PimaIndiansDiabetes dataset from the mlbench package to build a logistic regression model that predicts whether subjects have diabetes or not based on a set of variables.

PimaIndiansDiabetes dataset includes 768 observations and 9 variables.

```
#install.packages("mlbench") #To install the data package
library(mlbench)
data(PimaIndiansDiabetes) #for calling the data from mlbench


head(PimaIndiansDiabetes,10)
```

```
##     pregnant glucose pressure triceps insulin mass pedigree age diabetes
## 1          6     148       72      35       0 33.6    0.627  50      pos
## 2          1      85       66      29       0 26.6    0.351  31      neg
## 3          8     183       64       0       0 23.3    0.672  32      pos
## 4          1      89       66      23      94 28.1    0.167  21      neg
## 5          0     137       40      35     168 43.1    2.288  33      pos
## 6          5     116       74       0       0 25.6    0.201  30      neg
## 7          3      78       50      32      88 31.0    0.248  26      pos
## 8         10     115        0       0       0 35.3    0.134  29      neg
## 9          2     197       70      45     543 30.5    0.158  53      pos
## 10         8     125       96       0       0  0.0    0.232  54      pos
```

```
#To see the rate of diabetes variable which is categorical
table <- table(PimaIndiansDiabetes$diabetes)
table

##
## neg pos
## 500 268
```

Descriptive Statistics of the Data

```
#summary() function gives you a simple summary (descriptive statistics) of
each of the variables on the PimaIndiansDiabetes data


summary(PimaIndiansDiabetes)

##     pregnant         glucose         pressure         triceps
##  Min.   : 0.000   Min.   :  0.0   Min.   :  0.00   Min.   : 0.00
##  1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 62.00   1st Qu.: 0.00
##  Median : 3.000   Median :117.0   Median : 72.00   Median :23.00
##  Mean   : 3.845   Mean   :120.9   Mean   : 69.11   Mean   :20.54
##  3rd Qu.: 6.000   3rd Qu.:140.2   3rd Qu.: 80.00   3rd Qu.:32.00
##  Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00


##     insulin           mass           pedigree           age        diabetes
##  Min.   :  0.0   Min.   : 0.00   Min.   :0.0780   Min.   :21.00  neg:500
##  1st Qu.:  0.0   1st Qu.:27.30   1st Qu.:0.2437   1st Qu.:24.00  pos:268
##  Median : 30.5   Median :32.00   Median :0.3725   Median :29.00
##  Mean   : 79.8   Mean   :31.99   Mean   :0.4719   Mean   :33.24
##  3rd Qu.:127.2   3rd Qu.:36.60   3rd Qu.:0.6262   3rd Qu.:41.00
##  Max.   :846.0   Max.   :67.10   Max.   :2.4200   Max.   :81.00
```

Logistic Regression Model

Response (Target) variable: Diabetes

Explanatory (Feature) variables: Pregnant, Glucose, Pressure, Triceps, Insülin, Mass, Pedigree, Age

Model:

$$\text{logit}(p) = -8.4046964 + 0.1231823\mathbf{Pregnant} + 0.0351637\mathbf{Glucose} - 0.0132955\mathbf{Pressure} + 0.0006190\mathbf{Triceps} - 0.0011917\mathbf{Insulin} + 0.0897010\mathbf{Mass} + 0.9451797\mathbf{Pedigree} + 0.0148690\mathbf{Age}$$

```r
#Use the glm () function to create a logistic regression model

#Family function refers to the level of the response variable.

model <- glm(diabetes ~ pregnant + glucose + pressure + triceps + insulin +
mass + pedigree + age, data = PimaIndiansDiabetes, family = "binomial")
summary(model)
```

```
##
## Call:
## glm(formula = diabetes ~ pregnant + glucose + pressure + triceps +
##     insulin + mass + pedigree + age, family = "binomial", data = PimaInd
iansDiabetes)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5566  -0.7274  -0.4159   0.7267   2.9297
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -8.4046964  0.7166359 -11.728  < 2e-16 ***
## pregnant     0.1231823  0.0320776   3.840 0.000123 ***
## glucose      0.0351637  0.0037087   9.481  < 2e-16 ***
## pressure    -0.0132955  0.0052336  -2.540 0.011072 *
## triceps      0.0006190  0.0068994   0.090 0.928515
## insulin     -0.0011917  0.0009012  -1.322 0.186065
## mass         0.0897010  0.0150876   5.945 2.76e-09 ***
## pedigree     0.9451797  0.2991475   3.160 0.001580 **
## age          0.0148690  0.0093348   1.593 0.111192
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 993.48  on 767  degrees of freedom
## Residual deviance: 723.45  on 759  degrees of freedom
## AIC: 741.45
##
## Number of Fisher Scoring iterations: 5
```

Hypothesis tests on individual regression coefficients:

$H_0 = \beta_j = 0$

$H_1 = \beta_j \neq 0$

or

$H_0 =$ Variables have an effect on diabetes.

$H_1 =$ Variables have no effect on diabetes.

Looking at the summary, we can see which variables are significant by comparing the p-values.

Pregnant, glucose, pressure, mass and pedigree variables having a significant effect on the response variable (diabetes).

Triceps, insulin and age variables do not have a significant effect on the response variable (diabetes).

Interpretation of Odds Ratio

```
exp(coef(model)) #For odds ratio
```

```
##  (Intercept)    pregnant    glucose     pressure     triceps
## 0.0002238137 1.1310905981 1.0357892688 0.9867924485 1.0006191560
##   insulin       mass        pedigree       age
## 0.9988090108 1.0938471417 2.5732758592 1.0149800983
```

If the OR is <1, odds are decreased for an outcome; OR >1 means the odds are increased for a given outcome.

- One unit increase in **pregnant** variable increases the probability of having diabetes by 1.13 times.
- One unit increase in **glucose** variable increases the probability of having diabetes by 1.03 times.
- One unit increase in **triceps** variable increases the probability of having diabetes by 1 times.
- One unit increase in **mass** variable increases the probability of having diabetes by 1.09 times.
- One unit increase in **age** variable increases the probability of having diabetes by 1.01 times.
- One unit increase in **pedigree** variable increases the probability of having diabetes by 2.57 times.
- One-unit increase in the **pressure** variable decreases the probability of having diabetes by 0.98 times.
- One-unit increase in the **insulin** variable decreases the probability of having diabetes by 0.99 times.

Splitting Pimaindiansdiabetes Data

The following code splits 80% of the data selected randomly into training set and the remaining 20% sample into test set.

Training set is implemented to build up a model, while a test set is to validate the model built.

```r
set.seed(123)
index <- sample(nrow(PimaIndiansDiabetes), nrow(PimaIndiansDiabetes) * 0.8)
train <- PimaIndiansDiabetes[index,]
test  <- PimaIndiansDiabetes[-index,]

table(train$diabetes) #For imbalanced problem control

##
## neg pos
## 398 216

table(test$diabetes)

##
## neg pos
## 102  52
```

The number of negative observations is higher than the number of positive observations. In this case, there may be an imbalance problem.

Train a LRM on Train Set

```r
model1 <- glm(diabetes ~ pregnant + glucose + pressure + triceps + insulin
+ mass + pedigree + age, data = train, family = "binomial")
summary(model1)

##
## Call:
## glm(formula = diabetes ~ pregnant + glucose + pressure + triceps +
##     insulin + mass + pedigree + age, family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4798  -0.7300  -0.4281   0.7420   2.9493
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -8.2130001  0.7785903 -10.549  < 2e-16 ***
## pregnant     0.1183929  0.0357879   3.308 0.000939 ***
## glucose      0.0352638  0.0041847   8.427  < 2e-16 ***
## pressure    -0.0130875  0.0057262  -2.286 0.022281 *
## triceps     -0.0009585  0.0075277  -0.127 0.898678
## insulin     -0.0009091  0.0009840  -0.924 0.355526
## mass         0.0860673  0.0166329   5.175 2.29e-07 ***
## pedigree     0.7810790  0.3211506   2.432 0.015010 *
## age          0.0152288  0.0102926   1.480 0.138982
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##  Null deviance: 796.42  on 613  degrees of freedom
```

```
## Residual deviance: 583.55  on 605  degrees of freedom
## AIC: 601.55
##
## Number of Fisher Scoring iterations: 5
```

Performance of The Model on Train And Test Set

The accuracy rate is used to interpret the performance of the model.

Logistic regression returns probability values (between 0 and 1) as the response variable.

If the probabilities are less than 0.5, it is assigned as negative, if greater than 0.5, it is assigned as positive.

```
predicted_probs_train <- predict(model,train,type ="response" )
```

```
predicted_class_train <- ifelse ( predicted_probs_train > 0.5,"pos","neg" )
head(predicted_class_train,10)
```

```
##  415   463   179   526   195   118   299   229   244    14
## "neg" "neg" "pos" "neg" "neg" "neg" "neg" "pos" "neg" "pos"
```

```
#gives the value of accuracy
```

```
mean(predicted_class_train==train$diabetes)
```

```
## [1] 0.7801303
```

According to the result, the response variables in the train data are classified correctly by 78%.

```
predicted_probs_test <- predict(model1,test,type ="response" )
predicted_class_test <- ifelse ( predicted_probs_test > 0.5,"pos","neg" )
head(predicted_class_test,10)
```

```
##   1     3     9    17    22    27    28    32    42    43
## "pos" "pos" "pos" "neg" "neg" "pos" "neg" "pos" "pos" "neg"
```

```
#gives the value of accuracy
```

```
mean(predicted_class_test == test$diabetes)
```

```
## [1] 0.7922078
```

According to the result, the response variables in the test data are classified correctly by 79%

The train set accuracy is 0.78, while the test set accuracy is 0.79. Therefore, there might be a underfitting problem.

Underfittig Problem – Solution :

- Get more training data.
- Increase the size or number of parameters in the model.
- Increase the complexity of the model.

Confusion Matrix

```
table(predicted = predicted_class_test, actual=test$diabetes)
```

```
##               actual
## predicted   neg     pos
##       neg  92(TN)  22(FN)
##       pos  10(FP)  30(TP)
```

True negatives (TN) are the number of negative instances the classifier correctly identified as negative.

False negatives (FN) are, the number of instances classified as negative but in reality, are positive.

False positives (FP) are the number of instances in which the classifier identified as positive but in reality, are negative.

True positives (TP) are the number of positive instances the classifier correctly identified as positive.

- TP and TN are the correct guesses. A good classifier should have large TP and TN and small (ideally zero) numbers for FP and FN.
- FN and FP are large. This may be due to an imbalanced classification problem.
- The model mispredicted the positive because it learned the negative better.

```
table(test$diabetes) #For imbalanced problem control
```

```
##
##  neg↑ pos ↓
##  102  52
```

$$\textbf{Accurancy} = \frac{\text{TN} + \text{TP}}{\text{TN} + \text{FN} + \text{FP} + \text{TP}} = \frac{92 + 30}{92 + 22 + 10 + 30} = \frac{122}{154} = 0.79$$

$$\textbf{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{30}{30 + 22} = 0.57$$

$$\textbf{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}} = \frac{92}{92 + 10} = 0.90$$