Classification with Logistic Regression Tutorial

In this tutorial, we will learn how to evaluate the predictive ability of a logistic regression model, via confusion matrices, the ROC curve, and the AUC.

We will continue to use the `students.txt` dataset from the previous tutorial, that contains information on about 250 college students at a large public university and their study and party habits. The variables are:

- `Gender`: gender of student
- `Smoke`: whether the student smokes
- `Marijuan`: whether the student uses marijuana
- `DrivDrnk`: whether the student has driven while drunk
- `GPA`: student's GPA
- `PartyNum`: number of times the student parties in a month
- `DaysBeer`: number of days the student drinks at least 2 beers in a month
- `StudyHrs`: number of hours the students studies in a week

Suppose we want to relate the likelihood of a student driving while drunk with the other variables.

Let us read the data in:

```r
library(tidyverse)
Data<-read.table("students.txt", header=T, sep="")
```

We are going to perform some basic data wrangling for our dataframe:

- Remove the first column, as it is just an index.
- Keep observations that have no missing values in any variable. There are about a dozen observations with missing values in at least one variable.
- Apply `factor()` to categorical variables. As a reminder, this should be done to categorical variables if you want to change the reference class.

```r
##first column is index, remove it
Data<-Data[,-1]
##some NAs in data. Remove them
Data<-Data[complete.cases(Data),]

##convert categorical to factors. needed for contrasts
Data$Gender<-factor(Data$Gender)
Data$Smoke<-factor(Data$Smoke)
Data$Marijuan<-factor(Data$Marijuan)
Data$DrivDrnk<-factor(Data$DrivDrnk)
```

We are going to split the dataset into equal portions: one a training set, and another a test set. Recall that the training set is used to build the model, and the test set is used to assess how the model performs on new observations. We use the `set.seed()` function so that we can replicate the same split each time this block of code is run. An integer needs to be supplied to the `set.seed()` function.

```r
##set seed so results (split) are reproducible
set.seed(6021)

##evenly split data into train and test sets
sample.data<-sample.int(nrow(Data), floor(.50*nrow(Data)), replace = F)
```

```
train<-Data[sample.data, ]
test<-Data[-sample.data, ]
```

The `set.seed()` function allows us to have reproducible results. When we randomly split the data into a training and test set, we will get the same split each time we run this block of code.

The `sample.int()` function allows us to sample a vector of random integers.

- The first value is the maximum value of the integer we want to randomly generate, which in this case, will be the number of observations in our data frame.
- The second value represents the number of random integers we want to generate. In this case, this will be 50% of the number of observations, since we want a 50-50 split for the training and test data.
- The last value, `replace=F`, means we want to sample without replacement, that means once an integer is sampled, it cannot be sampled again.

Recall from the previous tutorial that we should use three predictors, `Smoke`, `Marijuan`, and `DaysBeer` instead of all of them. So we fit this logistic regression:

```
##fit model
reduced<-glm(DrivDrnk~Smoke+Marijuan+DaysBeer, family=binomial, data=train)
```

1 Confusion Matrices

To create a confusion matrix, we need to find the predicted probabilities for our test data, then use the `table()` function to tabulate the number of true positives, true negatives, false positives, and false negatives via a confusion matrix:

```
##predicted probs for test data
preds<-predict(reduced,newdata=test, type="response")

##confusion matrix with threshold of 0.5
table(test$DrivDrnk, preds>0.5)
```

```
##
##        FALSE TRUE
##   No      46   12
##   Yes     22   39
```

I have placed the actual response in the rows, and whether the predicted probability is greater than the threshold of 0.5 (the classification based on the model) in the columns.

For our model with a threshold of 0.5, we have:

- 46 observations that truly have not driven drunk and were correctly classified as not having driven drunk (true negative).
- 12 observations that truly have not driven drunk but were incorrectly classified as having driven drunk (false positive).
- 22 observations that truly have driven drunk and were correctly classified as having driven drunk (true positive).
- 39 observations that truly have driven drunk but were incorrectly classified as not having driven drunk (false negative).

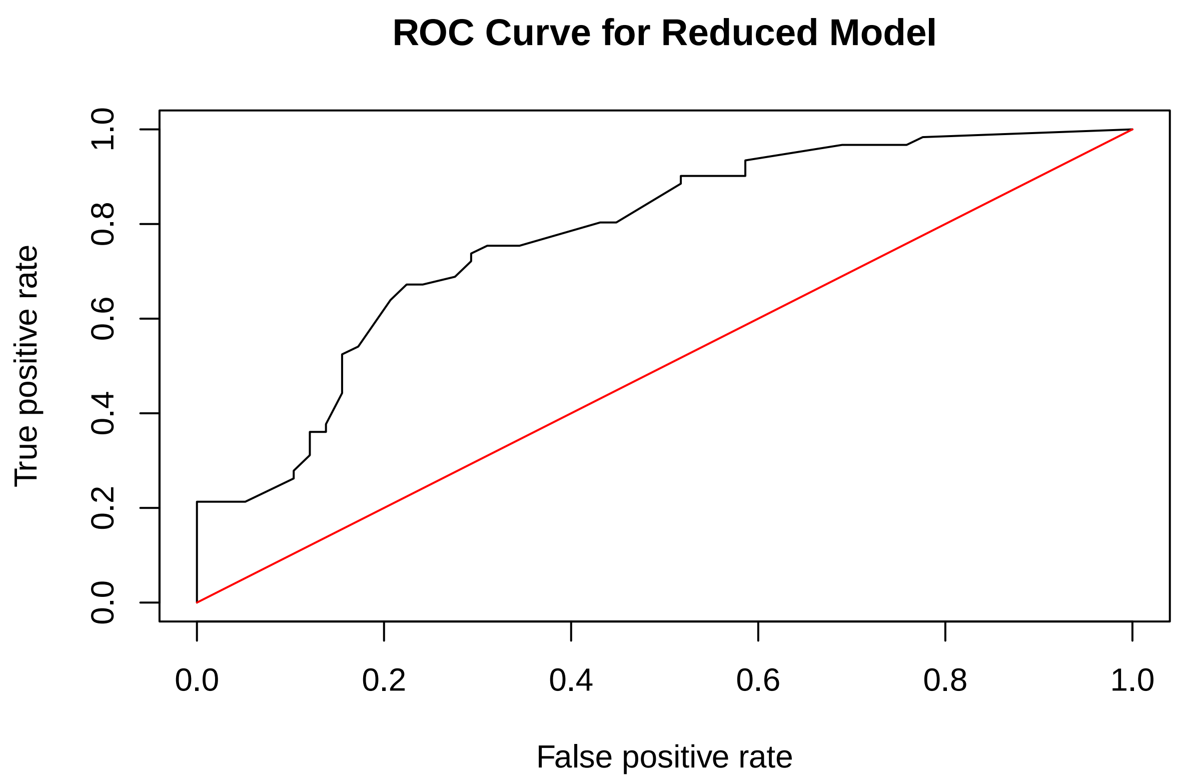
2 ROC Curve

The ROC curve plots the true positive rate (TPR) against the false positive rate (FPR) of the logistic regression as the threshold is varied from 0 to 1. We need a couple of functions from the `ROCR` package, so we load it and use the `prediction()` and `performance()` functions from it to create the ROC curve:

```r
library(ROCR)
##produce the numbers associated with classification table
rates<-ROCR::prediction(preds, test$DrivDrnk)

##store the true positive and false postive rates
roc_result<-ROCR::performance(rates,measure="tpr", x.measure="fpr")

##plot ROC curve and overlay the diagonal line for random guessing
plot(roc_result, main="ROC Curve for Reduced Model")
lines(x = c(0,1), y = c(0,1), col="red")
```



- The `prediction()` function transforms the vector of estimated probabilities and the vector of the response for the test set into a format that can be used with the `performance()` function to create the ROC curve.
- The object `roc_result` stores the values of the true positive rate and false positive rate via the `performance()` function. The arguments `measure` and `x.measure` are used to plot the true positive rate on the y-axis and false positive rate on the x-axis respectively.
- The function `lines()` is used to overlay the diagonal line on the plot for ease of comparison between random guessing and our model's classification ability.

As a reminder, the ROC curve gives us all possible combinations of TPRs and FPRs as the threshold is varied from 0 to 1. A perfect classification will result in a curve that has a TPR of 1 and FPR of 0, i.e. it will appear on the top left of the plot.

The red diagonal line represents a classifier that randomly guesses the binary outcome without using any information from the predictors. An ROC curve that is above this diagonal indicates the logistic regression does better than random guessing; an ROC curve that is below this diagonal does worse than random guessing.

So we can see that our model does better than a classifier that randomly guesses.

3 AUC

Another measure is the AUC. As its name suggests, it is simply the area under the (ROC) curve. A perfect classifier will have an AUC of 1. A classifier that randomly guesses will have an AUC of 0.5. Thus, AUCs closer to 1 are desirable. To obtain the AUC of our ROC

```r
##compute the AUC
auc<-ROCR::performance(rates, measure = "auc")
auc@y.values
```

```
## [[1]]
## [1] 0.7741662
```

The AUC of our ROC curve is 0.7742, which means our logistic regression does better than random guessing.

4 Practice

On your own, fit a logistic regression using all the predictors. Then obtain the confusion matrix with a threshold of 0.5, as well as the corresponding ROC curve and AUC of this model. Compare these with what we obtained for the model with just three predictors: `Smoke`, `Marijuan`, and `DaysBeer`. Which model does better based on these? Is the answer surprising?