Logistic Regression II

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

We will continue to use the `titanic.txt` data set that was used in the previous tutorial. As a reminder, the data set consists of data on some of the passengers on the Titanic passenger liner that sank on April 15, 1912. The variables are

- `Survived`: 0 for did not survive, 1 for survived
- `Sex`: gender of the passenger
- `Age`: age of the passenger
- `Fare`: fare paid by the passenger

We will use `Survived` as the response variable, and see if the odds of survival on the Titanic is associated with the other predictors.

Read the data in, and install and load the `ROCR` package. We will be using functions from the `ROCR` package to create ROC curves and evaluate the AUC associated with our logistic regression.

```
library(ROCR)
Data<-read.table("titanic.txt", header=TRUE, sep="")
```

A common way to validate data is to randomly split our data into two portions: one portion being the training data, which is used to build the model, and the other portion is the test data, which is used to evaluate how well our model does in predicting new observations. To randomly split the data

```
##set the random number generator so same results can
##be reproduced
set.seed(111)

##choose the observations to be in the training set.
##I am splitting the data set into halves
sample<-sample.int(nrow(Data), floor(.50*nrow(Data)), replace = F)
train<-Data[sample, ] ##training data
test<-Data[-sample, ] ##test 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.

1) ROC curve

Now that we have our training data, we can fit a logistic regression

```
full<-glm(Survived~., family=binomial, data=train)
summary(full)
```

```
##
## Call:
## glm(formula = Survived ~ ., family = binomial, data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2568  -0.6313  -0.5620   0.8300   1.9681
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.474870   0.333829   1.422 0.154883
## Sexmale     -2.373289   0.270475  -8.775  < 2e-16 ***
## Age          0.001127   0.009170   0.123 0.902149
## Fare         0.013117   0.003666   3.578 0.000346 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 473.30  on 352  degrees of freedom
## Residual deviance: 351.24  on 349  degrees of freedom
## AIC: 359.24
##
## Number of Fisher Scoring iterations: 5
```

Only 1 predictor, `Age`, is insignificant, so we can drop it based on the Wald test. Refit the model without `Age`

```
result<-glm(Survived~Sex+Fare, family="binomial", data=train)
summary(result)
```

```
##
## Call:
## glm(formula = Survived ~ Sex + Fare, family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2706  -0.6283  -0.5629   0.8309   1.9631
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.50424    0.23349   2.160  0.03081 *
## Sexmale     -2.36877    0.26780  -8.845  < 2e-16 ***
## Fare         0.01316    0.00365   3.606  0.00031 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 473.30  on 352  degrees of freedom
## Residual deviance: 351.25  on 350  degrees of freedom
## AIC: 357.25
##
## Number of Fisher Scoring iterations: 5
```

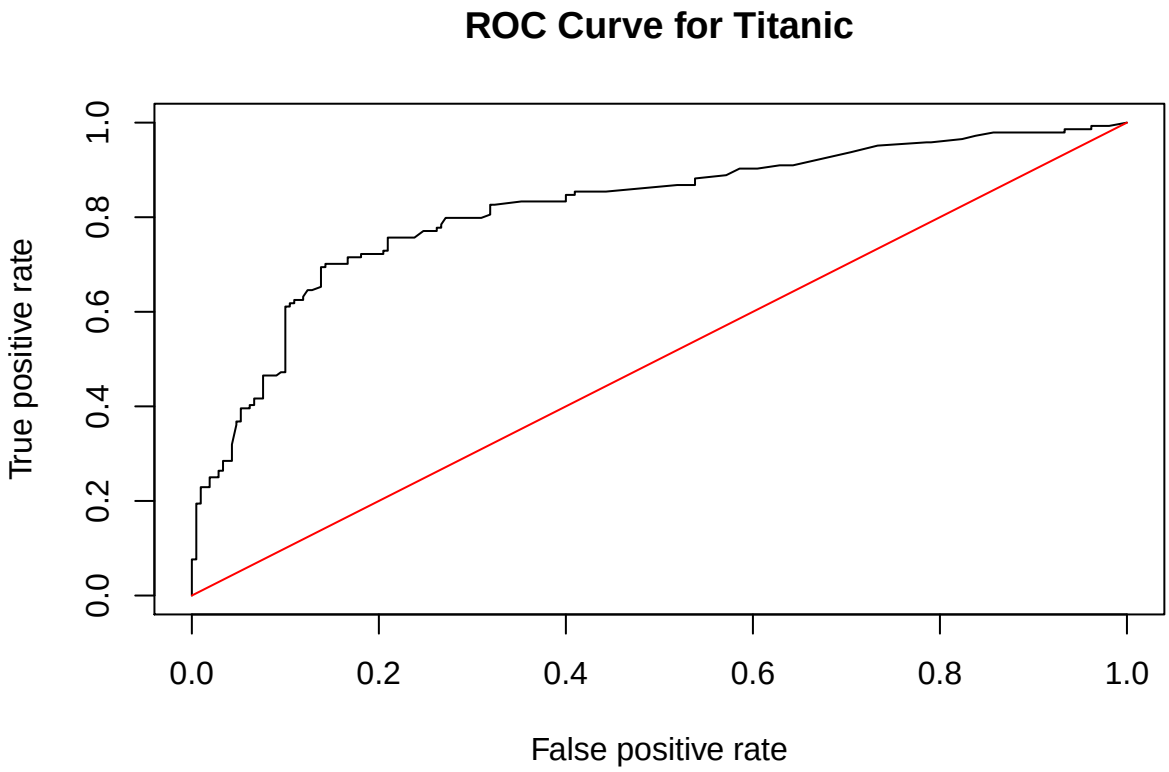
We are now ready to create the ROC curve for our logistic regression. The ROC plots the true positive rate (TPR) against the false positive rate (FPR) of the logistic regression as the threshold value is varied from 0 to 1.

```
##predicted survival rate for test data based on training data
preds<-predict(result,newdata=test, type="response")

##transform the input data into a format that is suited for the
##performance() function
rates<-prediction(preds, test$Survived)

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

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



- The object `preds` stores the predicted probabilities of the test data based on our logistic regression.
- The `prediction()` function transforms the input data into a format that can be used with the `performance()` function to create the ROC curve. It requires the predicted probabilities of the test data, as well as the true values of the test data.
- 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.
- Note: the `prediction()` and `performance()` functions come from the `ROCR` package.

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; and ROC curve that is below this diagonal does worse than random guessing.

2) AUC

Another measure 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

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

```
## [[1]]
## [1] 0.8184193
```

The AUC of our ROC curve is 0.8184, which means our logistic regression does better than random guessing.

3) Confusion matrices

While ROC curves and the AUC inform us how our logistic regression performs as the threshold is varied, these measures do not inform us about the TPR and FPR for specific thresholds.

To create a confusion matrix, we need to specify a value for the threshold, and then create a table containing the number of observations in each class, and the number of predicted observations in each class (for the test data). For example, we want a confusion matrix where the threshold is 0.5

```
table(test$Survived, preds>0.5)
```

```
##
##     FALSE TRUE
##   0   180   30
##   1    44  100
```

From this table, we note the following:

- The first row represents the number of observations that did not survive, which is 210.
- Out of the 210 that did not survive, 180 were predicted not to survive, while 30 were (wrongly) predicted to survive. So the FPR is $\frac{30}{210} = 0.14$.
- The second row represents the number of observations that survived, which is 144.
- Out of the 144 that survived, 44 were (wrongly) predicted to not survive, while 100 were predicted to survive. So the TPR is $\frac{100}{144} = 0.69$.

In some contexts, we may be willing to tradeoff between TPR and FPR, and so we can adjust the threshold accordingly. For example, to lower the FPR, we can raise the threshold to 0.7 (raising the threshold makes it more difficult for an observation to be classified as a survivor)

```
table(test$Survived, preds>0.7)
```

```
##
##     FALSE TRUE
##   0   201    9
##   1    98   46
```

For a threshold of 0.7, the FPR is $\frac{9}{210} = 0.04$ and the TPR is $\frac{46}{144} = 0.32$. Notice we an higher threshold, the FPR is reduced, but is done at the expense of lowering the TPR as well.