SurvMeth 640 Assignment 1

Cheng, Chia Wen

2023-02-06

Setup

```
library(titanic)
library(caret)
library(pROC)
library(ggplot2)
library(ggmap)
library(e1071)
library(class)
library(PRROC)
library(ROSE)
```

Data

In this notebook we use the Titanic data that is used on Kaggle (https://www.kaggle.com) as an introductory competition for getting familiar with machine learning. It includes information on a set of Titanic passengers, such as age, sex, ticket class and whether he or she survived the Titanic tragedy.

Source: https://www.kaggle.com/c/titanic/data

```
titanic <- titanic_train
str(titanic)
```

```
## 'data.frame':    891 obs. of  12 variables:
##  $ PassengerId: int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Survived   : int  0 1 1 1 0 0 0 0 1 1 ...
##  $ Pclass     : int  3 1 3 1 3 3 1 3 3 2 ...
##  $ Name       : chr  "Braund, Mr. Owen Harris" "Cumings, Mrs. John Bradley (Florence Briggs Thayer)"
##  $ Sex        : chr  "male" "female" "female" "female" ...
##  $ Age        : num  22 38 26 35 35 NA 54 2 27 14 ...
##  $ SibSp      : int  1 1 0 1 0 0 0 3 0 1 ...
##  $ Parch      : int  0 0 0 0 0 0 0 1 2 0 ...
##  $ Ticket     : chr  "A/5 21171" "PC 17599" "STON/O2. 3101282" "113803" ...
##  $ Fare       : num  7.25 71.28 7.92 53.1 8.05 ...
##  $ Cabin      : chr  "" "C85" "" "C123" ...
##  $ Embarked   : chr  "S" "C" "S" "S" ...
```

We begin with some minor data preparations. The `lapply()` function is a handy tool if the task is to apply the same transformation (e.g. `as.factor()`) to multiple columns of a data frame.

```
titanic[, c(2:3,5,12)] <- lapply(titanic[, c(2:3,5,12)], as.factor)
```

The `age` variable has some NAs, as a quick and dirty solution we can create a categorized age variable with NAs as an additional factor level.

```
titanic$Age_c <- cut(titanic$Age, 5)
titanic$Age_c <- addNA(titanic$Age_c)
summary(titanic$Age_c)
```

```
## (0.34,16.3] (16.3,32.3] (32.3,48.2] (48.2,64.1] (64.1,80.1]        <NA>
##         100         346         188          69          11         177
```

Train and test set

Next we split the data into a training (80%) and a test (20%) part. This can be done by random sampling with `sample()`.

```
set.seed(9395)
train <- sample(1:nrow(titanic), 0.8*nrow(titanic))
c_train <- titanic[train,]
c_test <- titanic[-train,]

X_train <- titanic[train, c(1, 3, 5, 10, 12, 13)]
X_test <- titanic[-train, c(1, 3, 5, 10, 12, 13)]
y_test <- titanic[-train, c(1, 2)] # survived testing data
```

Logistic regression

In this exercise we simply use logistic regression as our prediction method, since we want to focus on the evaluation part. Build a first logit model with `Survived` as the outcome and `Pclass`, `Sex`, `Age_c`, `Fare` and `Embarked` as features.

```
mod_logit_1 <- glm(Survived ~ Pclass + Sex + Age_c + Fare + Embarked, data = c_train, family = "binomia
summary(mod_logit_1)
```

```
##
## Call:
## glm(formula = Survived ~ Pclass + Sex + Age_c + Fare + Embarked,
##     family = "binomial", data = c_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5542  -0.6624  -0.3512   0.6283   2.3946
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)       1.799e+01  1.692e+03   0.011  0.99152
## Pclass2          -8.664e-01  3.287e-01  -2.636  0.00839 **
## Pclass3          -2.272e+00  3.241e-01  -7.009  2.4e-12 ***
## Sexmale          -2.543e+00  2.128e-01 -11.952  < 2e-16 ***
## Age_c(16.3,32.3] -8.498e-01  3.321e-01  -2.559  0.01050 *
## Age_c(32.3,48.2] -1.232e+00  3.758e-01  -3.279  0.00104 **
## Age_c(48.2,64.1] -1.579e+00  5.008e-01  -3.152  0.00162 **
## Age_c(64.1,80.1] -1.696e+01  6.977e+02  -0.024  0.98061
## Age_cNA          -1.178e+00  3.800e-01  -3.102  0.00193 **
## Fare             -1.111e-04  2.203e-03  -0.050  0.95978
## EmbarkedC        -1.417e+01  1.692e+03  -0.008  0.99332
## EmbarkedQ        -1.399e+01  1.692e+03  -0.008  0.99340
## EmbarkedS        -1.475e+01  1.692e+03  -0.009  0.99304
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 950.78  on 711  degrees of freedom
## Residual deviance: 624.22  on 699  degrees of freedom
## AIC: 650.22
##
## Number of Fisher Scoring iterations: 15
```

A quick look at the coefficients of the first logit model.

```
coef(mod_logit_1)
```

```
##      (Intercept)          Pclass2          Pclass3          Sexmale
##     1.798949e+01    -8.663790e-01    -2.271831e+00    -2.543134e+00
## Age_c(16.3,32.3] Age_c(32.3,48.2] Age_c(48.2,64.1] Age_c(64.1,80.1]
##    -8.498386e-01    -1.232365e+00    -1.578757e+00    -1.695617e+01
##          Age_cNA             Fare        EmbarkedC        EmbarkedQ
##    -1.178496e+00    -1.110793e-04    -1.416856e+01    -1.399430e+01
##        EmbarkedS
##    -1.474986e+01
```

Now, build an additional logit model that uses the same features, but includes at least one interaction or non-linear term.

```
mod_logit_2 <- glm(Survived ~ Pclass + Sex + Age_c + Fare + Embarked + Age_c*Fare, data = c_train, famil
summary(mod_logit_2)
```

```
##
## Call:
## glm(formula = Survived ~ Pclass + Sex + Age_c + Fare + Embarked +
##     Age_c * Fare, family = "binomial", data = c_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2189  -0.6146  -0.3658   0.5915   2.5289
##
## Coefficients:
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)             1.845e+01  1.679e+03   0.011 0.991234
## Pclass2                -1.003e+00  3.436e-01  -2.920 0.003501 **
## Pclass3               -2.485e+00  3.487e-01  -7.128 1.02e-12 ***
## Sexmale               -2.583e+00  2.168e-01 -11.914  < 2e-16 ***
## Age_c(16.3,32.3]      -1.132e+00  4.421e-01  -2.561 0.010445 *  
## Age_c(32.3,48.2]      -1.991e+00  5.141e-01  -3.872 0.000108 ***
## Age_c(48.2,64.1]      -1.934e+00  6.856e-01  -2.821 0.004794 ** 
## Age_c(64.1,80.1]      -1.621e+01  1.350e+03  -0.012 0.990416    
## Age_cNA               -1.400e+00  4.983e-01  -2.810 0.004955 ** 
## Fare                  -1.289e-02  8.519e-03  -1.513 0.130212    
## EmbarkedC             -1.405e+01  1.679e+03  -0.008 0.993324    
## EmbarkedQ             -1.392e+01  1.679e+03  -0.008 0.993384    
## EmbarkedS             -1.459e+01  1.679e+03  -0.009 0.993066    
## Age_c(16.3,32.3]:Fare  7.953e-03  8.933e-03   0.890 0.373289    
## Age_c(32.3,48.2]:Fare  1.905e-02  9.625e-03   1.980 0.047727 *  
## Age_c(48.2,64.1]:Fare  9.712e-03  1.081e-02   0.898 0.369008    
## Age_c(64.1,80.1]:Fare -2.477e-02  3.918e+01  -0.001 0.999496    
## Age_cNA:Fare           5.698e-03  1.068e-02   0.533 0.593785    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 950.78  on 711  degrees of freedom
## Residual deviance: 616.39  on 694  degrees of freedom
## AIC: 652.39
## 
## Number of Fisher Scoring iterations: 15
```

I think Age may have interaction with Fare because the elder tend to have saved more money or are earning more than the younger people. In addition, older people may emphasize their need of physical comfortness so that they are able to relax. Thus, I added the interaction term between factoral Age in groups and numeric Fare.

Again, summarize the resulting object.

```
coef(mod_logit_2)
```

```
##           (Intercept)               Pclass2               Pclass3 
##          18.446520943          -1.003200950          -2.485105588 
##               Sexmale      Age_c(16.3,32.3]      Age_c(32.3,48.2] 
##          -2.583179952          -1.132085881          -1.990757580 
##      Age_c(48.2,64.1]      Age_c(64.1,80.1]               Age_cNA 
##          -1.933850208         -16.213087051          -1.400085830 
##                  Fare             EmbarkedC             EmbarkedQ 
##          -0.012891822         -14.048659948         -13.923269250 
##             EmbarkedS Age_c(16.3,32.3]:Fare Age_c(32.3,48.2]:Fare 
##         -14.592246160           0.007952965           0.019054789 
## Age_c(48.2,64.1]:Fare Age_c(64.1,80.1]:Fare          Age_cNA:Fare 
##           0.009712021          -0.024769430           0.005698375
```

Prediction in test set

Given both logit objects, we can generate predicted risk scores/ predicted probabilities of `Survived` in the test set.

```
yp_logit_1 <- predict(mod_logit_1, newdata = c_test, type = "response")
yp_logit_1.fac <- as.factor(ifelse(yp_logit_1 > 0.5, "1", "0"))
yp_logit_1.fac
```

```
##   1   2   3   4   5   7  11  12  15  20  25  31  33  37  38  39  40  53  58  65
##   0   1   1   1   0   0   1   1   1   1   1   1   1   0   0   1   1   1   0   1
##  72  73  76  78  85  87  91 103 109 134 139 146 150 158 167 168 169 172 174 175
##   1   0   0   0   1   0   0   0   0   1   0   0   0   0   1   0   0   0   0   0
## 193 195 198 204 205 229 234 236 238 242 252 254 258 263 264 269 271 273 276 277
##   1   1   0   0   0   0   1   0   1   1   1   0   1   0   0   1   0   1   1   0
## 278 287 297 309 314 316 318 338 339 351 353 355 362 374 376 379 380 384 385 390
##   0   0   0   0   0   1   0   1   0   0   0   0   0   1   1   0   0   1   0   1
## 396 397 398 400 410 413 418 419 435 436 443 445 446 450 453 454 461 482 493 504
##   0   1   0   1   0   1   1   0   0   1   0   0   1   0   1   0   0   0   0   0
## 509 516 525 526 529 530 544 550 552 553 556 557 563 565 576 587 588 591 592 594
##   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   1   1
## 599 605 608 613 620 621 624 626 629 631 638 641 642 646 647 655 662 664 665 668
##   0   1   0   1   0   0   0   0   0   0   0   0   1   1   0   1   0   0   0   0
## 669 670 676 685 702 710 712 715 723 727 745 750 751 755 757 759 767 773 775 783
##   0   1   0   0   0   0   0   0   0   1   0   0   1   1   0   0   1   1   1   0
## 786 798 801 804 805 818 832 835 837 838 841 842 850 851 866 870 879 882 883
##   0   1   0   0   0   0   0   0   0   0   0   0   1   0   1   0   0   0   1
## Levels: 0 1
```

```
yp_logit_2 <- predict(mod_logit_2, newdata = c_test, type = "response")
yp_logit_2.fac<- as.factor(ifelse(yp_logit_2 > 0.5, "1", "0"))
yp_logit_2.fac
```

```
##   1   2   3   4   5   7  11  12  15  20  25  31  33  37  38  39  40  53  58  65
##   0   1   1   1   0   0   1   1   1   1   1   0   1   0   0   1   1   1   0   1
##  72  73  76  78  85  87  91 103 109 134 139 146 150 158 167 168 169 172 174 175
##   1   0   0   0   1   0   0   0   0   1   0   0   0   0   1   0   0   0   0   0
## 193 195 198 204 205 229 234 236 238 242 252 254 258 263 264 269 271 273 276 277
##   1   1   0   0   0   0   1   0   1   1   1   0   1   0   0   1   0   1   1   0
## 278 287 297 309 314 316 318 338 339 351 353 355 362 374 376 379 380 384 385 390
##   0   0   0   0   0   1   0   1   0   0   0   0   0   1   1   0   0   1   0   1
## 396 397 398 400 410 413 418 419 435 436 443 445 446 450 453 454 461 482 493 504
##   0   1   0   1   0   1   1   0   0   1   0   0   1   0   1   0   0   0   0   0
## 509 516 525 526 529 530 544 550 552 553 556 557 563 565 576 587 588 591 592 594
##   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   1   1
## 599 605 608 613 620 621 624 626 629 631 638 641 642 646 647 655 662 664 665 668
##   0   0   0   1   0   0   0   0   0   0   0   0   1   1   0   1   0   0   0   0
## 669 670 676 685 702 710 712 715 723 727 745 750 751 755 757 759 767 773 775 783
##   0   1   0   0   0   0   0   0   0   1   0   0   1   1   0   0   1   1   1   0
## 786 798 801 804 805 818 832 835 837 838 841 842 850 851 866 870 879 882 883
##   0   1   0   0   0   0   1   0   0   0   0   1   1   0   1   0   0   0   1
## Levels: 0 1
```

It is often useful to first get an idea of prediction performance independent of specific classification thresholds. Use the `pROC` (or `PRROC`) package to create roc objects for both risk score vectors.

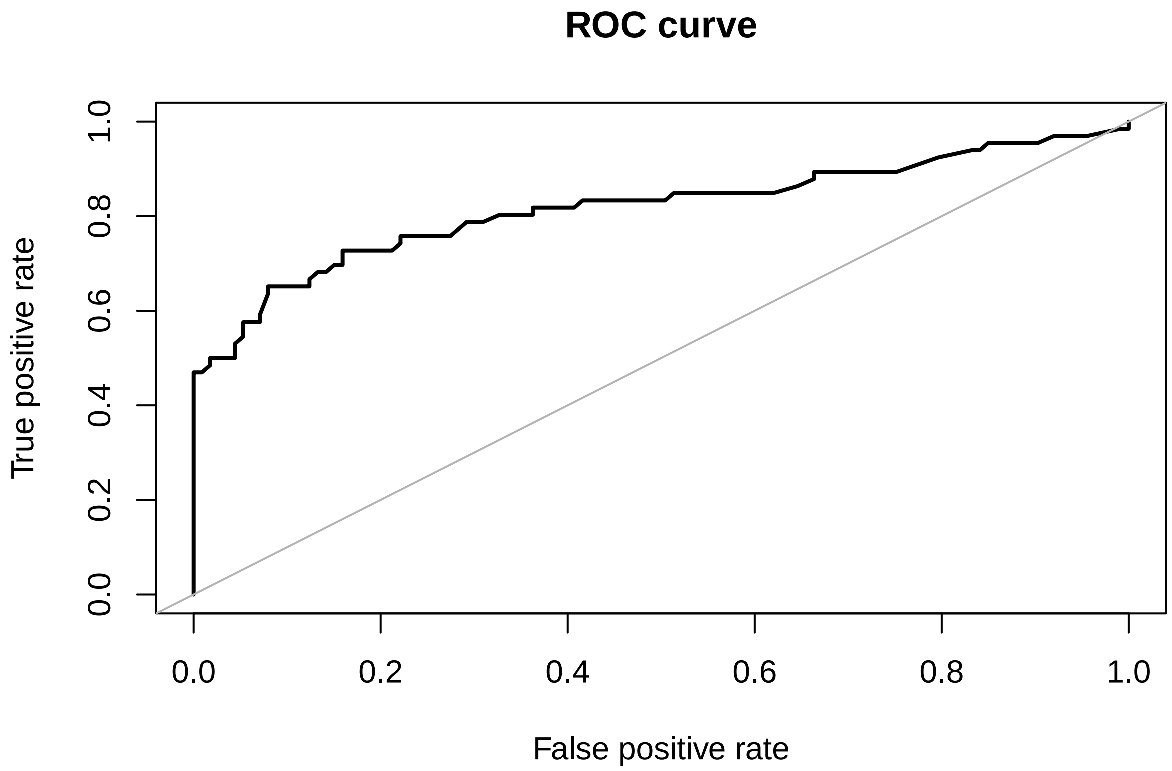
```r
nc_1 <- yp_logit_1.fac[y_test$Survived == "0"]
nc_1
```

```
##   1   5   7  15  25  31  38  39  58  65  72  73  76  78  87  91 103 109 139 146 
##   0   0   0   1   1   1   0   1   0   1   1   0   0   0   0   0   0   0   0   0 
## 150 158 168 169 172 174 175 198 204 229 236 252 254 263 264 271 277 278 297 309 
##   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0 
## 314 318 351 353 355 362 374 379 380 385 396 397 398 410 419 435 443 453 482 493 
##   0   0   0   0   0   0   1   0   0   0   0   1   0   0   0   0   0   1   0   0 
## 504 509 516 525 526 529 530 552 553 556 563 565 576 587 591 594 599 620 621 624 
##   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0 
## 626 629 638 641 647 655 662 664 668 669 676 685 712 715 723 750 757 759 767 773 
##   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   1   1 
## 783 786 801 818 835 837 838 841 842 851 879 882 883 
##   0   0   0   0   0   0   0   0   0   0   0   0   1 
## Levels: 0 1
```

```r
pc_1 <- yp_logit_1.fac[y_test$Survived == "1"]
pc_1
```

```
##   2   3   4  11  12  20  33  37  40  53  85 134 167 193 195 205 234 238 242 258 
##   1   1   1   1   1   1   1   0   1   1   1   1   1   1   1   0   1   1   1   1 
## 269 273 276 287 316 338 339 376 384 390 400 413 418 436 445 446 450 454 461 544 
##   1   1   1   0   1   1   0   1   1   1   1   1   1   1   0   1   0   0   0   0 
## 550 557 588 592 605 608 613 631 642 646 665 670 702 710 727 745 751 755 775 798 
##   0   1   0   1   1   0   1   0   1   1   0   1   0   0   1   0   1   1   1   1 
## 804 805 832 850 866 870 
##   0   0   0   1   1   0 
## Levels: 0 1
```

```r
roc_1 <- roc.curve(y_test$Survived, yp_logit_1, plotit = TRUE, add.roc = FALSE)
```



```
roc_1
```

```
## Area under the curve (AUC): 0.818
```

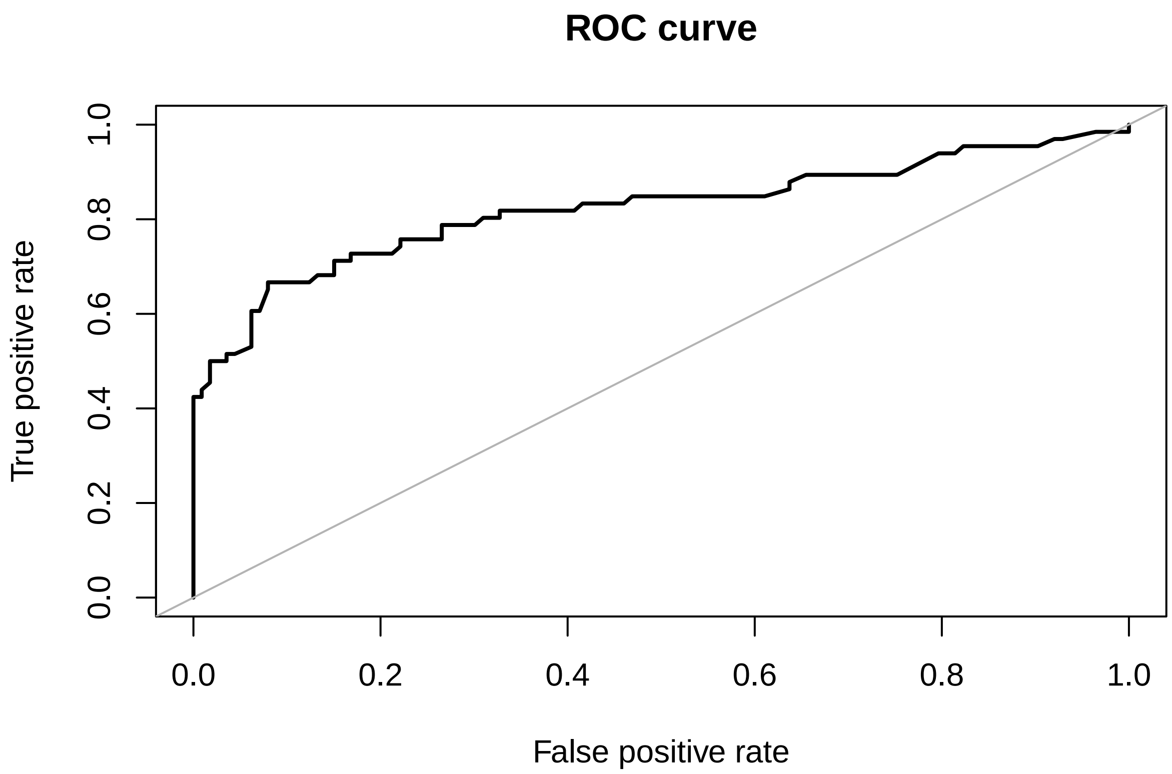
```
nc_2 <- yp_logit_2.fac[y_test$Survived == "0"]
nc_2
```

```
##   1   5   7  15  25  31  38  39  58  65  72  73  76  78  87  91 103 109 139 146
##   0   0   0   1   1   0   0   1   0   1   1   0   0   0   0   0   0   0   0   0
## 150 158 168 169 172 174 175 198 204 229 236 252 254 263 264 271 277 278 297 309
##   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0   0   0   0   0
## 314 318 351 353 355 362 374 379 380 385 396 397 398 410 419 435 443 453 482 493
##   0   0   0   0   0   0   1   0   0   0   0   1   0   0   0   0   0   1   0   0
## 504 509 516 525 526 529 530 552 553 556 563 565 576 587 591 594 599 620 621 624
##   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   1   0   0   0   0
## 626 629 638 641 647 655 662 664 668 669 676 685 712 715 723 750 757 759 767 773
##   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   1   1
## 783 786 801 818 835 837 838 841 842 851 879 882 883
##   0   0   0   0   0   0   0   0   1   0   0   0   1
## Levels: 0 1
```

```
pc_2 <- yp_logit_2.fac[y_test$Survived == "1"]
pc_2
```

```
##   2   3   4  11  12  20  33  37  40  53  85 134 167 193 195 205 234 238 242 258
##   1   1   1   1   1   1   1   0   1   1   1   1   1   1   1   0   1   1   1   1
## 269 273 276 287 316 338 339 376 384 390 400 413 418 436 445 446 450 454 461 544
##   1   1   1   0   1   1   0   1   1   1   1   1   1   1   0   1   0   0   0   0
## 550 557 588 592 605 608 613 631 642 646 665 670 702 710 727 745 751 755 775 798
##   0   1   0   1   0   0   1   0   1   1   0   1   0   0   1   0   1   1   1   1
## 804 805 832 850 866 870
##   0   0   1   1   1   0
## Levels: 0 1
```

```
roc_2 <- roc.curve(y_test$Survived, yp_logit_2, plotit = TRUE, add.roc = FALSE)
```



```
roc_2
```

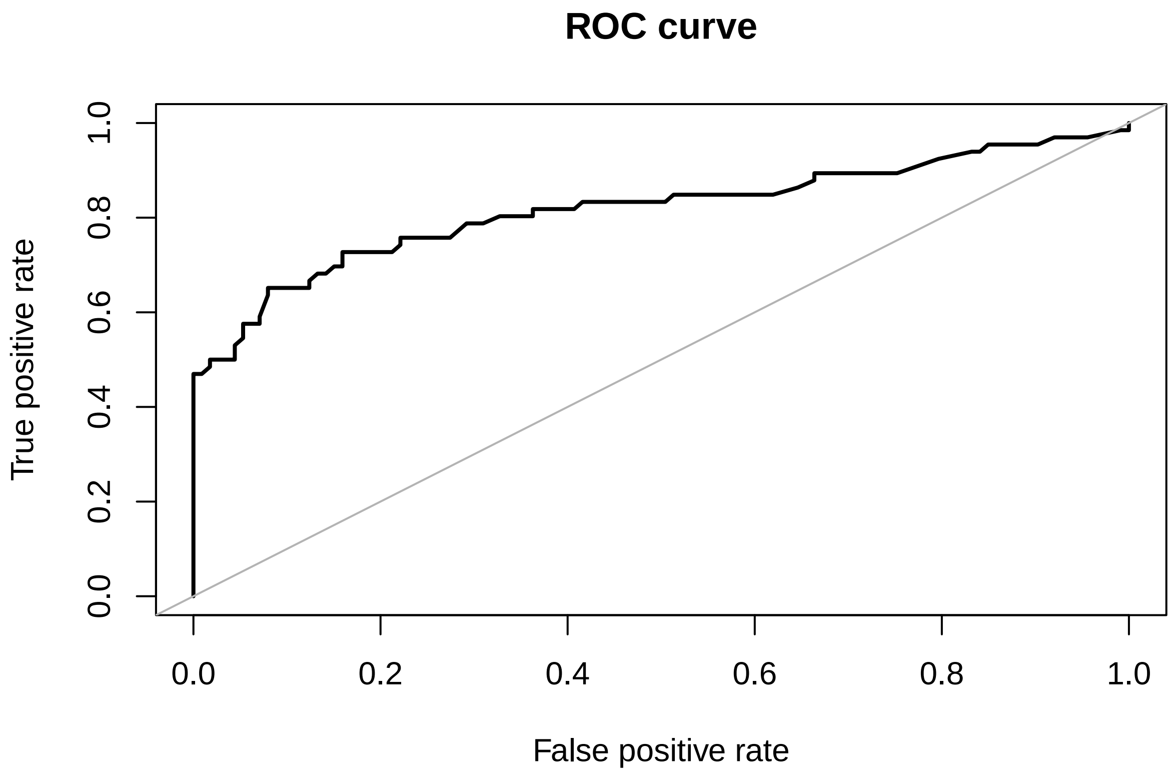
```
## Area under the curve (AUC): 0.822
```

Neither pROC nor PRROC packages worked for me; thus, I used ROSE as an alternative option. But I put the lines below.

```
## using the package 'PRROC' but not getting the assumed correct answers of the AUCs
# detach("package:ROSE", unload = TRUE)
# roc_1try <- roc.curve(scores.class0 = pc_1, scores.class1 = nc_1, curve = T)
# roc_1try
# roc_2try <- roc.curve(scores.class0 = pc_2, scores.class1 = nc_2, curve = T)
# roc_2try
## they turned out with exactly the same AUC value of 0.7745374, however, I've made sure everything att
```

Now, you can print and plot the resulting `roc` objects.

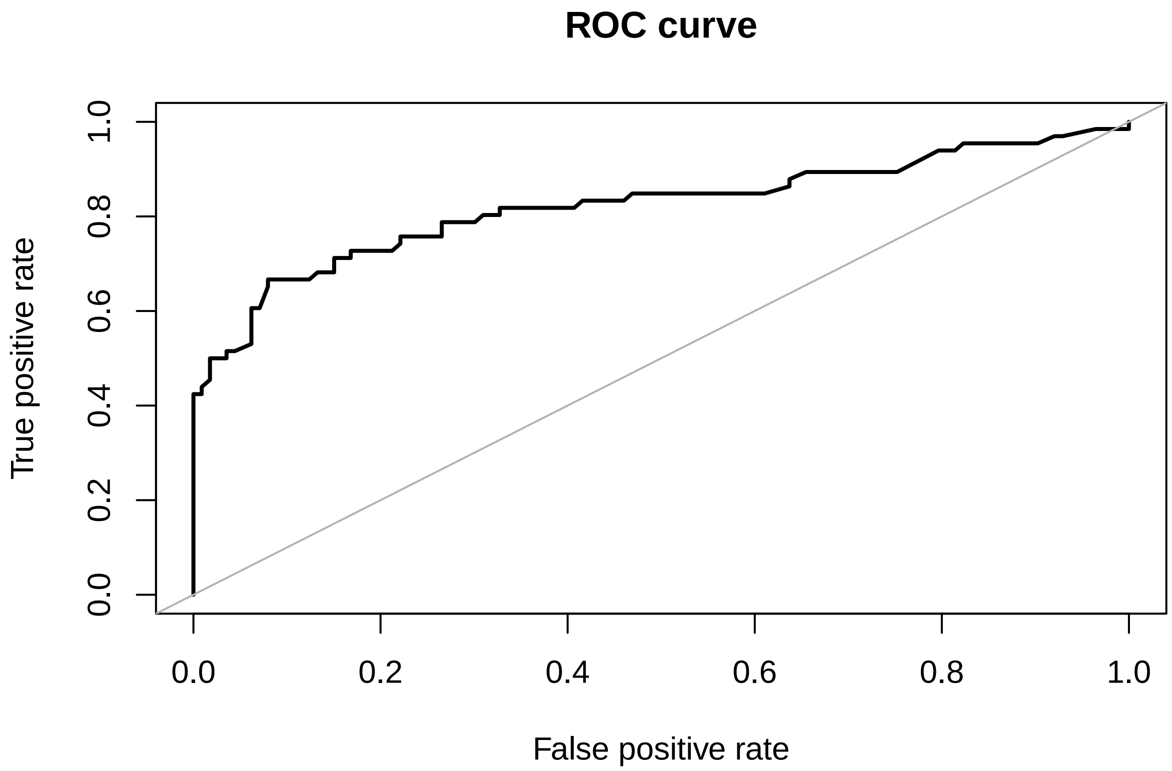
```
library(ROSE)
roc_1 <- roc.curve(y_test$Survived, yp_logit_1, plotit = TRUE, add.roc = FALSE)
```



```
roc_1
```

```
## Area under the curve (AUC): 0.818
```

```
roc_2 <- roc.curve(y_test$Survived, yp_logit_2, plotit = TRUE, add.roc = FALSE)
```



```
roc_2
```

```
## Area under the curve (AUC): 0.822
```

```
## using the package 'PRROC' but not getting the assumed correct plots and AUCs
# plot(roc_1try, scale.color = heat.colors(100))
# plot(roc_2try, scale.color = heat.colors(100))
## since they turned out with exactly the same number of 0.7745374, the two plot looked exactly the sam
```

In your own words, how would you interpret these ROC curves? What do you think about the ROC-AUCs we observe here?

From the output of the two models using package ROSE (since it works over PRROC), the area under the curves are both greater than 0.8, indicating well predictions of the two logistic models. Furthermore, the non-linear prediction model, which includes an interaction term in the model, performed better than the one without interaction terms, since it has a slightly greater AUC.

end

As a next step, we want to predict class membership given the risk scores of our two models. Here we use the default classification threshold, 0.5.

```
yp_logit_1.fac <- as.factor(ifelse(yp_logit_1 > 0.5, "1", "0"))
yp_logit_2.fac <- as.factor(ifelse(yp_logit_2 > 0.5, "1", "0"))
```

On this basis, we can use `confusionMatrix()` to get some performance measures for the predicted classes.

```
yp_logit_1.fac <- as.factor(ifelse(yp_logit_1 > 0.5, "TRUE", "FALSE"))
yp_logit_2.fac <- as.factor(ifelse(yp_logit_2 > 0.5, "TRUE", "FALSE"))
y_test$Survived_TF <- as.factor(ifelse(y_test$Survived == 1, "TRUE", "FALSE"))
confusionMatrix(yp_logit_1.fac, y_test$Survived_TF, mode = "everything", positive = "TRUE")
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction FALSE TRUE
##      FALSE    98   21
##      TRUE     15   45
##                                          
##                Accuracy : 0.7989         
##                  95% CI : (0.7326, 0.855)
##     No Information Rate : 0.6313         
##     P-Value [Acc > NIR] : 9.106e-07      
##                                          
##                   Kappa : 0.5597         
##                                          
##  Mcnemar's Test P-Value : 0.4047         
##                                          
##             Sensitivity : 0.6818         
##             Specificity : 0.8673         
##          Pos Pred Value : 0.7500         
##          Neg Pred Value : 0.8235         
##               Precision : 0.7500         
##                  Recall : 0.6818         
##                      F1 : 0.7143         
##              Prevalence : 0.3687         
##          Detection Rate : 0.2514         
##    Detection Prevalence : 0.3352         
##       Balanced Accuracy : 0.7745         
##                                          
##        'Positive' Class : TRUE           
## 
```

```
confusionMatrix(yp_logit_2.fac, y_test$Survived_TF, mode = "everything", positive = "TRUE")
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction FALSE TRUE
##      FALSE    98   21
##      TRUE     15   45
##                                          
##                Accuracy : 0.7989         
##                  95% CI : (0.7326, 0.855)
##      No Information Rate : 0.6313
##      P-Value [Acc > NIR] : 9.106e-07
##
##                    Kappa : 0.5597
##
##   Mcnemar's Test P-Value : 0.4047
##
##              Sensitivity : 0.6818
##              Specificity : 0.8673
##           Pos Pred Value : 0.7500
##           Neg Pred Value : 0.8235
##                Precision : 0.7500
##                   Recall : 0.6818
##                       F1 : 0.7143
##               Prevalence : 0.3687
##           Detection Rate : 0.2514
##     Detection Prevalence : 0.3352
##        Balanced Accuracy : 0.7745
##
##         'Positive' Class : TRUE
##
```

Briefly explain potential limitations when measuring prediction performance as carried out in the last two code chunks.

Since the four different prediction elements in *yp_logit_1.fac* and *yp_logit_2.fac* are reversed with the same counts of "FALSE" and "TRUE", the results are similar. Both models have levels of accuracy at almost 0.8, inticating good performances. A potential limitation is that Logistic regression assumes linearity between the dependent variable and the predictors.

end