SMM636 Machine Learning (PRD2 A 2019/20)

R exercises 6: kNN versus LDA and more on model assessment

DR. Rui Zhu

2020-02-09

In R exercises 6, you will know:

- How to calculate performance measures, such as sensitivity and specificity
- How to write self-defined functions
- How to get ROC curves

Don't forget to change your working directory!

Performance measures

The `Caravan` data have 85 predictors that measure demographic characteristics for 5,822 individuals. The response variable is `Purchase`, which indicates whether or not a given individual purchases a caravan insurance policy. {Note that in this dataset, only 6% people purchase a caravan insurance policy. The two classes are very imbalanced!}

This data is part of the `ISLR` library. To use this dataset:

```r
library(ISLR)
summary(Caravan)
```

```
##     MOSTYPE        MAANTHUI        MGEMOMV         MGEMLEEF
##  Min.   : 1.00   Min.   : 1.000   Min.   :1.000   Min.   :1.000
##  1st Qu.:10.00   1st Qu.: 1.000   1st Qu.:2.000   1st Qu.:2.000
##  Median :30.00   Median : 1.000   Median :3.000   Median :3.000
##  Mean   :24.25   Mean   : 1.111   Mean   :2.679   Mean   :2.991
##  3rd Qu.:35.00   3rd Qu.: 1.000   3rd Qu.:3.000   3rd Qu.:3.000
##  Max.   :41.00   Max.   :10.000   Max.   :5.000   Max.   :6.000
##     MOSHOOFD         MGODRK           MGODPR          MGODOV
##  Min.   : 1.000   Min.   :0.0000   Min.   :0.000   Min.   :0.00
##  1st Qu.: 3.000   1st Qu.:0.0000   1st Qu.:4.000   1st Qu.:0.00
##  Median : 7.000   Median :0.0000   Median :5.000   Median :1.00
##  Mean   : 5.774   Mean   :0.6965   Mean   :4.627   Mean   :1.07
##  3rd Qu.: 8.000   3rd Qu.:1.0000   3rd Qu.:6.000   3rd Qu.:2.00
##  Max.   :10.000   Max.   :9.0000   Max.   :9.000   Max.   :5.00
##      MGODGE          MRELGE          MRELSA           MRELOV
##  Min.   :0.000   Min.   :0.000   Min.   :0.0000   Min.   :0.00
##  1st Qu.:2.000   1st Qu.:5.000   1st Qu.:0.0000   1st Qu.:1.00
##  Median :3.000   Median :6.000   Median :1.0000   Median :2.00
##  Mean   :3.259   Mean   :6.183   Mean   :0.8835   Mean   :2.29
##  3rd Qu.:4.000   3rd Qu.:7.000   3rd Qu.:1.0000   3rd Qu.:3.00
##  Max.   :9.000   Max.   :9.000   Max.   :7.0000   Max.   :9.00
##     MFALLEEN        MFGEKIND       MFWEKIND      MOPLHOOG
##  Min.   :0.000   Min.   :0.00   Min.   :0.0   Min.   :0.000
##  1st Qu.:0.000   1st Qu.:2.00   1st Qu.:3.0   1st Qu.:0.000
##  Median :2.000   Median :3.00   Median :4.0   Median :1.000
```

```
##  Mean   :1.888   Mean   :3.23   Mean   :4.3   Mean   :1.461
##  3rd Qu.:3.000   3rd Qu.:4.00   3rd Qu.:6.0   3rd Qu.:2.000
##  Max.   :9.000   Max.   :9.00   Max.   :9.0   Max.   :9.000
##     MOPLMIDD        MOPLLAAG        MBERHOOG        MBERZELF
##  Min.   :0.000   Min.   :0.000   Min.   :0.000   Min.   :0.000
##  1st Qu.:2.000   1st Qu.:3.000   1st Qu.:0.000   1st Qu.:0.000
##  Median :3.000   Median :5.000   Median :2.000   Median :0.000
##  Mean   :3.351   Mean   :4.572   Mean   :1.895   Mean   :0.398
##  3rd Qu.:4.000   3rd Qu.:6.000   3rd Qu.:3.000   3rd Qu.:1.000
##  Max.   :9.000   Max.   :9.000   Max.   :9.000   Max.   :5.000
##     MBERBOER         MBERMIDD        MBERARBG        MBERARBO
##  Min.   :0.0000   Min.   :0.000   Min.   :0.00   Min.   :0.000
##  1st Qu.:0.0000   1st Qu.:2.000   1st Qu.:1.00   1st Qu.:1.000
##  Median :0.0000   Median :3.000   Median :2.00   Median :2.000
##  Mean   :0.5223   Mean   :2.899   Mean   :2.22   Mean   :2.306
##  3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:3.00   3rd Qu.:3.000
##  Max.   :9.0000   Max.   :9.000   Max.   :9.00   Max.   :9.000
##       MSKA           MSKB1           MSKB2            MSKC
##  Min.   :0.000   Min.   :0.000   Min.   :0.000   Min.   :0.000
##  1st Qu.:0.000   1st Qu.:1.000   1st Qu.:1.000   1st Qu.:2.000
##  Median :1.000   Median :2.000   Median :2.000   Median :4.000
##  Mean   :1.621   Mean   :1.607   Mean   :2.203   Mean   :3.759
##  3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu.:3.000   3rd Qu.:5.000
##  Max.   :9.000   Max.   :9.000   Max.   :9.000   Max.   :9.000
##       MSKD           MHHUUR          MHKOOP          MAUT1
##  Min.   :0.000   Min.   :0.000   Min.   :0.000   Min.   :0.00
##  1st Qu.:0.000   1st Qu.:2.000   1st Qu.:2.000   1st Qu.:5.00
##  Median :1.000   Median :4.000   Median :5.000   Median :6.00
##  Mean   :1.067   Mean   :4.237   Mean   :4.772   Mean   :6.04
##  3rd Qu.:2.000   3rd Qu.:7.000   3rd Qu.:7.000   3rd Qu.:7.00
##  Max.   :9.000   Max.   :9.000   Max.   :9.000   Max.   :9.00
##      MAUT2           MAUT0          MZFONDS          MZPART
##  Min.   :0.000   Min.   :0.000   Min.   :0.000   Min.   :0.000
##  1st Qu.:0.000   1st Qu.:1.000   1st Qu.:5.000   1st Qu.:1.000
##  Median :1.000   Median :2.000   Median :7.000   Median :2.000
##  Mean   :1.316   Mean   :1.959   Mean   :6.277   Mean   :2.729
##  3rd Qu.:2.000   3rd Qu.:3.000   3rd Qu.:8.000   3rd Qu.:4.000
##  Max.   :7.000   Max.   :9.000   Max.   :9.000   Max.   :9.000
##     MINKM30         MINK3045        MINK4575        MINK7512
##  Min.   :0.000   Min.   :0.000   Min.   :0.000   Min.   :0.0000
##  1st Qu.:1.000   1st Qu.:2.000   1st Qu.:1.000   1st Qu.:0.0000
##  Median :2.000   Median :4.000   Median :3.000   Median :0.0000
##  Mean   :2.574   Mean   :3.536   Mean   :2.731   Mean   :0.7961
##  3rd Qu.:4.000   3rd Qu.:5.000   3rd Qu.:4.000   3rd Qu.:1.0000
##  Max.   :9.000   Max.   :9.000   Max.   :9.000   Max.   :9.0000
##     MINK123M         MINKGEM         MKOOPKLA        PWAPART
##  Min.   :0.0000   Min.   :0.000   Min.   :1.000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:3.000   1st Qu.:0.0000
##  Median :0.0000   Median :4.000   Median :4.000   Median :0.0000
##  Mean   :0.2027   Mean   :3.784   Mean   :4.236   Mean   :0.7712
##  3rd Qu.:0.0000   3rd Qu.:4.000   3rd Qu.:6.000   3rd Qu.:2.0000
##  Max.   :9.0000   Max.   :9.000   Max.   :8.000   Max.   :3.0000
##     PWABEDR           PWALAND           PPERSAUT       PBESAUT
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.00   Min.   :0.00000
```

```
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00   1st Qu.:0.00000
##  Median :0.00000   Median :0.00000   Median :5.00   Median :0.00000
##  Mean   :0.04002   Mean   :0.07162   Mean   :2.97   Mean   :0.04827
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:6.00   3rd Qu.:0.00000
##  Max.   :6.00000   Max.   :4.00000   Max.   :8.00   Max.   :7.00000
##     PMOTSCO          PVRAAUT            PAANHANG         PTRACTOR
##  Min.   :0.0000   Min.   :0.000000   Min.   :0.00000   Min.   :0.00000
##  1st Qu.:0.0000   1st Qu.:0.000000   1st Qu.:0.00000   1st Qu.:0.00000
##  Median :0.0000   Median :0.000000   Median :0.00000   Median :0.00000
##  Mean   :0.1754   Mean   :0.009447   Mean   :0.02096   Mean   :0.09258
##  3rd Qu.:0.0000   3rd Qu.:0.000000   3rd Qu.:0.00000   3rd Qu.:0.00000
##  Max.   :7.0000   Max.   :9.000000   Max.   :5.00000   Max.   :6.00000
##      PWERKT            PBROM          PLEVEN          PPERSONG
##  Min.   :0.00000   Min.   :0.000   Min.   :0.0000   Min.   :0.00000
##  1st Qu.:0.00000   1st Qu.:0.000   1st Qu.:0.0000   1st Qu.:0.00000
##  Median :0.00000   Median :0.000   Median :0.0000   Median :0.00000
##  Mean   :0.01305   Mean   :0.215   Mean   :0.1948   Mean   :0.01374
##  3rd Qu.:0.00000   3rd Qu.:0.000   3rd Qu.:0.0000   3rd Qu.:0.00000
##  Max.   :6.00000   Max.   :6.000   Max.   :9.0000   Max.   :6.00000
##     PGEZONG           PWAOREG            PBRAND         PZEILPL
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.000   Min.   :0.0000000
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.000   1st Qu.:0.0000000
##  Median :0.00000   Median :0.00000   Median :2.000   Median :0.0000000
##  Mean   :0.01529   Mean   :0.02353   Mean   :1.828   Mean   :0.0008588
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:4.000   3rd Qu.:0.0000000
##  Max.   :3.00000   Max.   :7.00000   Max.   :8.000   Max.   :3.0000000
##     PPLEZIER           PFIETS            PINBOED           PBYSTAND
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.00000   Min.   :0.00000
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000
##  Median :0.00000   Median :0.00000   Median :0.00000   Median :0.00000
##  Mean   :0.01889   Mean   :0.02525   Mean   :0.01563   Mean   :0.04758
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000
##  Max.   :6.00000   Max.   :1.00000   Max.   :6.00000   Max.   :5.00000
##     AWAPART         AWABEDR           AWALAND           APERSAUT
##  Min.   :0.000   Min.   :0.00000   Min.   :0.00000   Min.   :0.0000
##  1st Qu.:0.000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.0000
##  Median :0.000   Median :0.00000   Median :0.00000   Median :1.0000
##  Mean   :0.403   Mean   :0.01477   Mean   :0.02061   Mean   :0.5622
##  3rd Qu.:1.000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:1.0000
##  Max.   :2.000   Max.   :5.00000   Max.   :1.00000   Max.   :7.0000
##     ABESAUT           AMOTSCO           AVRAAUT            AAANHANG
##  Min.   :0.00000   Min.   :0.00000   Min.   :0.000000   Min.   :0.00000
##  1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.000000   1st Qu.:0.00000
##  Median :0.00000   Median :0.00000   Median :0.000000   Median :0.00000
##  Mean   :0.01048   Mean   :0.04105   Mean   :0.002233   Mean   :0.01254
##  3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.000000   3rd Qu.:0.00000
##  Max.   :4.00000   Max.   :8.00000   Max.   :3.000000   Max.   :3.00000
##     ATRACTOR           AWERKT              ABROM             ALEVEN
##  Min.   :0.00000   Min.   :0.000000   Min.   :0.00000   Min.   :0.00000
##  1st Qu.:0.00000   1st Qu.:0.000000   1st Qu.:0.00000   1st Qu.:0.00000
##  Median :0.00000   Median :0.000000   Median :0.00000   Median :0.00000
##  Mean   :0.03367   Mean   :0.006183   Mean   :0.07042   Mean   :0.07661
##  3rd Qu.:0.00000   3rd Qu.:0.000000   3rd Qu.:0.00000   3rd Qu.:0.00000
##  Max.   :4.00000   Max.   :6.000000   Max.   :2.00000   Max.   :8.00000
```

```
##     APERSONG          AGEZONG          AWAOREG           ABRAND
##  Min.   :0.000000   Min.   :0.000000   Min.   :0.000000   Min.   :0.0000
##  1st Qu.:0.000000   1st Qu.:0.000000   1st Qu.:0.000000   1st Qu.:0.0000
##  Median :0.000000   Median :0.000000   Median :0.000000   Median :1.0000
##  Mean   :0.005325   Mean   :0.006527   Mean   :0.004638   Mean   :0.5701
##  3rd Qu.:0.000000   3rd Qu.:0.000000   3rd Qu.:0.000000   3rd Qu.:1.0000
##  Max.   :1.000000   Max.   :1.000000   Max.   :2.000000   Max.   :7.0000
##     AZEILPL             APLEZIER            AFIETS
##  Min.   :0.0000000   Min.   :0.000000   Min.   :0.00000
##  1st Qu.:0.0000000   1st Qu.:0.000000   1st Qu.:0.00000
##  Median :0.0000000   Median :0.000000   Median :0.00000
##  Mean   :0.0005153   Mean   :0.006012   Mean   :0.03178
##  3rd Qu.:0.0000000   3rd Qu.:0.000000   3rd Qu.:0.00000
##  Max.   :1.0000000   Max.   :2.000000   Max.   :3.00000
##     AINBOED            ABYSTAND       Purchase
##  Min.   :0.000000   Min.   :0.00000   No :5474
##  1st Qu.:0.000000   1st Qu.:0.00000   Yes: 348
##  Median :0.000000   Median :0.00000
##  Mean   :0.007901   Mean   :0.01426
##  3rd Qu.:0.000000   3rd Qu.:0.00000
##  Max.   :2.000000   Max.   :2.00000
```

From `summary`, we can see that some variables have different scales.

To standardise the data:

```r
Caravan_scale=scale(Caravan [,-86])
```

Check the standard deviation and mean of the variables are all 1 and 0, respectively.

Create the training and test set and apply a kNN model with $k = 1$ to see the test result.

```r
# test sample index
index=1:1000
# get training and test set
train.X=Caravan_scale[-index,]
test.X=Caravan_scale[index,]
train.Y=Caravan$Purchase[-index]
test.Y=Caravan$Purchase[index]
# kNN
library("class")
set.seed (198)
knn.pred=knn(train.X,test.X,train.Y,k=1)
# mean of error rate
mean(test.Y!=knn.pred)
```

```
## [1] 0.118
```

The error rate is just around 12%, which is good. However, considering the imbalance feature of this data: we can get an error rate of 6% if we predict all test observations as No. Thus the error rate is no longer a good measure to assess the quality of the model. In this dataset, we care more about the accuracy of predicting people would like to buy the insurance. If without any prediction, we have to visit every customer to ask if they would like to buy and the success rate is just 6%. However, if we do our research first, then we could only visit customers who are likely to buy the insurance, which saves time and resources.

```r
table(knn.pred,test.Y)
```

```
##         test.Y
```

```
## knn.pred  No Yes
##      No  873  50
##      Yes  68   9
```

```r
9/(68+9)
```

```
## [1] 0.1168831
```

In our kNN model, we correctly predicted 9 customers who would buy the insurance and the accuracy is 11.7%, which is much larger than 6%. This shows the advantage of using kNN. Try different values of k to see the change of accuracy.

User-defined functions

To calculate other performance measures such as sensitivity and specificity, we can define our own function as follows.

```r
####################################################
#### This function calculates two performance measures:
#### specificity and sensitivity
#### Input: pred: predicted labels (factor)
####        truth: true labels (factor)
####        pos: positive level
####        neg: negative level
#### Output: a list containing sensitivity and specificity
####################################################
performance.measure<-function(pred,truth,pos,neg){
  #### get confusion table
  confusion=table(pred,truth)
  #### get tn, tp, fn, fp
  tn=confusion[neg,neg]
  tp=confusion[pos,pos]
  fn=confusion[neg,pos]
  fp=confusion[pos,neg]
  #### calculate sensitivity
  sens=tp/(tp+fn)
  #### calculate specificity
  spec=tn/(tn+fp)
  #### put the two values in a list
  measures=list(sensitivity=sens,specificity=spec)
  #### return the list as function output
  return(measures)
}
```

Save this script as `performance-measure.r` in your working directory.

Here is how to use this user-defined function:

```r
# kNN
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```r
set.seed(47)
knn3_pred=knn3Train(train.X, test.X, train.Y, k = 9, prob=TRUE)
```

```
#### calculate sensitivity and specificity
source("performance-measure.R")
pos=levels(test.Y)[2]; neg=levels(test.Y)[1]
measures=performance.measure(as.factor(knn3_pred),test.Y,pos,neg)
measures
```

```
## $sensitivity
## [1] 0.01694915
##
## $specificity
## [1] 1
##
## $accuracy
## [1] 0.942
```

We can compare the above result with the results from functions in the `caret` function.

```
sensitivity(as.factor(knn3_pred),test.Y,pos)
```

```
## [1] 0.01694915
```

```
specificity(as.factor(knn3_pred),test.Y,neg)
```

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

Exercise

Add the calculation of classification accuracy in `performance-measure.r` and return a list containing sensitivity, specificity and classification accuracy.

ROC curves

We can use the `ROCR` package to get ROC curves.

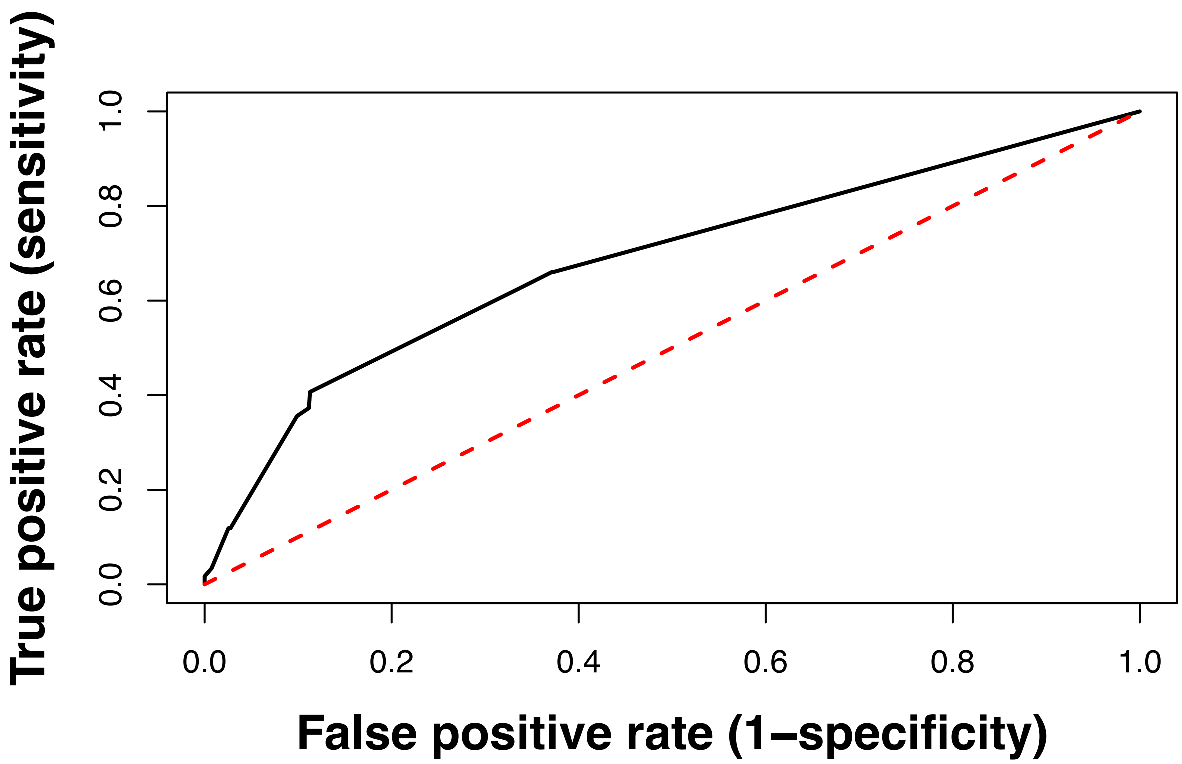
```
library(ROCR)
```

```
## Loading required package: gplots
```

```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

```
####################################################
#### ROC curve
att=attributes(knn3_pred)$prob
pred.ROCR = prediction(att[,2], (test.Y))
roc.curve = performance(pred.ROCR,"tpr","fpr")
plot(roc.curve,lwd=2,cex.lab=1.5,cex.axis=1.5,  font.lab=2,
     xlab="False positive rate (1-specificity)",
     ylab="True positive rate (sensitivity)")
#### add a line with auc=0.5
x=seq(0,1,0.01); y=x
lines(x,y,lwd =2, col =" red",lty=2)
```



We can also draw the ROC curves with the `pROC` package. Here we show how to do this with the models built in the `caret` package.

```
#### set up train control
fitControl <- trainControl(## 5-fold CV
  method = "repeatedcv",
  number = 5,
  ## repeated five times
  repeats = 5,
  summaryFunction = twoClassSummary,
  classProbs = TRUE)
#### training process
set.seed(5)
knnFit=train(train.X,train.Y, method = "knn",
              trControl = fitControl,
              metric = "ROC",
              preProcess = c("center","scale"),
              tuneLength=5)
knnFit
```

```
## k-Nearest Neighbors
##
## 4822 samples
##   85 predictor
##    2 classes: 'No', 'Yes'
##
## Pre-processing: centered (85), scaled (85)
## Resampling: Cross-Validated (5 fold, repeated 5 times)
## Summary of sample sizes: 3858, 3858, 3858, 3857, 3857, 3858, ...
## Resampling results across tuning parameters:
##
```

```
##    k   ROC        Sens       Spec
##     5  0.5896505  0.9928525  0.020036298
##     7  0.6103408  0.9973973  0.011058681
##     9  0.6189570  0.9990294  0.006218996
##    11  0.6289133  0.9997352  0.001379310
##    13  0.6346906  0.9999558  0.000000000
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was k = 13.
```

```r
knn.pred <- predict(knnFit,test.X)
confusionMatrix(knn.pred,test.Y)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  No Yes
##        No  941  59
##        Yes   0   0
##
##                Accuracy : 0.941
##                  95% CI : (0.9246, 0.9548)
##     No Information Rate : 0.941
##     P-Value [Acc > NIR] : 0.5346
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : 4.321e-14
##
##             Sensitivity : 1.000
##             Specificity : 0.000
##          Pos Pred Value : 0.941
##          Neg Pred Value :   NaN
##              Prevalence : 0.941
##          Detection Rate : 0.941
##    Detection Prevalence : 1.000
##       Balanced Accuracy : 0.500
##
##        'Positive' Class : No
##
```

Now we can draw an ROC curve to check the classification performance on test data.

```r
knn.probs <- predict(knnFit,test.X,type="prob")
head(knn.probs)
```

```
##          No        Yes
## 1 0.9230769 0.07692308
## 2 1.0000000 0.00000000
## 3 1.0000000 0.00000000
## 4 1.0000000 0.00000000
## 5 1.0000000 0.00000000
## 6 1.0000000 0.00000000
```

```r
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

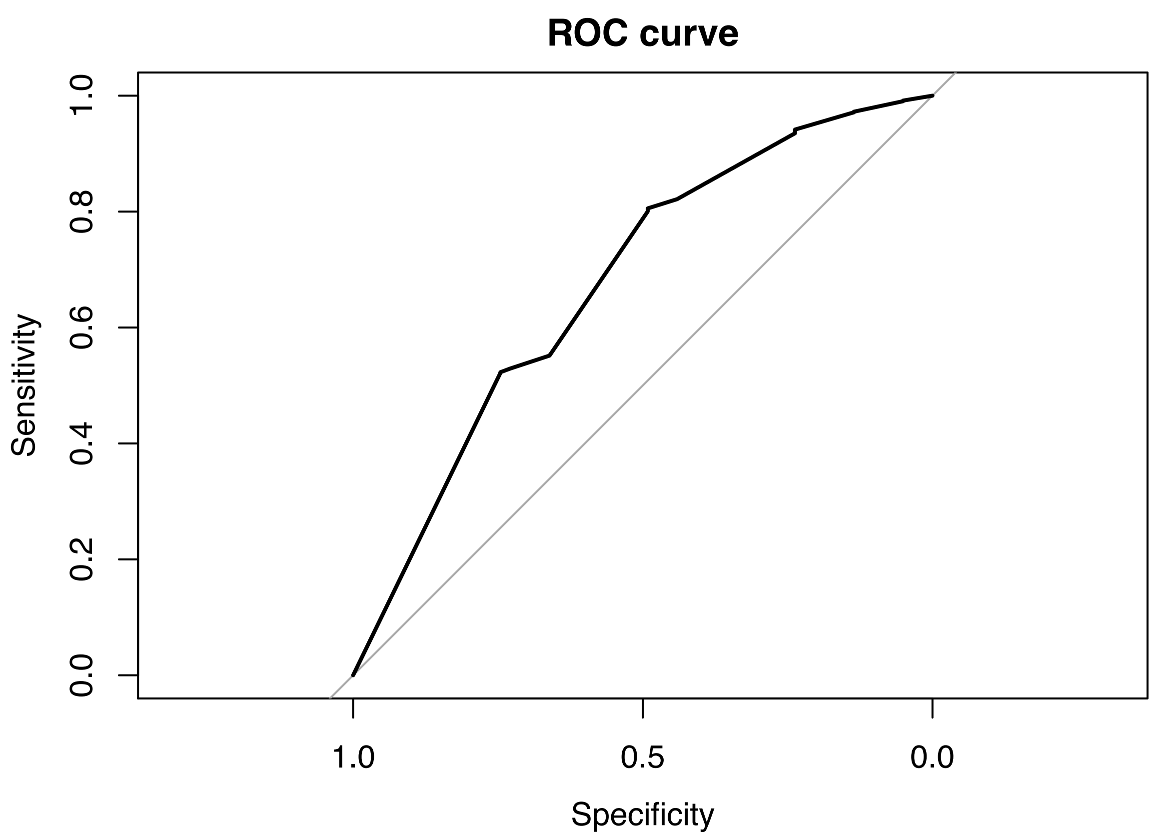
```
knn.ROC <- roc(predictor=knn.probs$No,
               response=test.Y,
               levels=rev(levels(test.Y)))
```

```
## Setting direction: controls < cases
```

```
knn.ROC$auc
```

```
## Area under the curve: 0.6775
```

```
plot(knn.ROC,main="ROC curve")
```



Assess the classification performances of kNN and LDA on the German Credit data

Classification on the original German Credit data

Load the German Credit data

```
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
data(GermanCredit)
## Delete two variables where all values are the same for both classes
GermanCredit[,c("Purpose.Vacation","Personal.Female.Single")] <- list(NULL)
```

Creat training/test split

```
# create training and test sets
set.seed(12)
trainIndex = createDataPartition(GermanCredit$Class, p = 0.7, list = FALSE, times = 1)
train.feature=GermanCredit[trainIndex,-10] # training features
train.label=GermanCredit$Class[trainIndex] # training labels
test.feature=GermanCredit[-trainIndex,-10] # test features
test.label=GermanCredit$Class[-trainIndex] # test labels
```

kNN

```
####################################
#### knn
#### set up train control
fitControl <- trainControl(## 5-fold CV
  method = "repeatedcv",
  number = 5,
  ## repeated five times
  repeats = 5,
  summaryFunction = twoClassSummary,
  classProbs = TRUE)
#### training process
set.seed(5)
knnFit=train(train.feature,train.label, method = "knn",
             trControl = fitControl,
             metric = "ROC",
             preProcess = c("center","scale"),
             tuneLength=10)
knnFit
```

```
## k-Nearest Neighbors
##
## 700 samples
##  59 predictor
##   2 classes: 'Bad', 'Good'
##
## Pre-processing: centered (59), scaled (59)
## Resampling: Cross-Validated (5 fold, repeated 5 times)
## Summary of sample sizes: 560, 560, 560, 560, 560, 560, ...
## Resampling results across tuning parameters:
##
##   k   ROC        Sens       Spec
##    5  0.6882556  0.3295238  0.8791837
##    7  0.7108017  0.3285714  0.9102041
##    9  0.7235666  0.3019048  0.9163265
##   11  0.7299028  0.2647619  0.9306122
##   13  0.7313703  0.2400000  0.9391837
##   15  0.7353596  0.2219048  0.9493878
##   17  0.7355053  0.1933333  0.9546939
##   19  0.7317833  0.1876190  0.9604082
##   21  0.7295092  0.1619048  0.9632653
##   23  0.7295773  0.1571429  0.9689796
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was k = 17.
```

LDA

```
####################################
#### lda
ldaFit=train(train.feature,train.label, method = "lda",
             trControl = trainControl(method = "none"))
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear
```

```
ldaFit$finalModel
```

```
## Call:
## lda(x, y)
##
## Prior probabilities of groups:
##  Bad Good
##  0.3  0.7
##
## Group means:
##      Duration   Amount InstallmentRatePercentage ResidenceDuration
## Bad  24.72381 4109.448                  3.119048          2.819048
## Good 18.63265 2852.037                  2.975510          2.851020
##           Age NumberExistingCredits NumberPeopleMaintenance Telephone
## Bad  33.55714              1.361905                1.142857 0.6380952
## Good 36.48571              1.424490                1.171429 0.5857143
##      ForeignWorker CheckingAccountStatus.lt.0
## Bad      0.9809524                  0.4571429
## Good     0.9469388                  0.2020408
```

```
##      CheckingAccountStatus.0.to.200 CheckingAccountStatus.gt.200
## Bad                        0.352381                   0.02380952
## Good                       0.244898                   0.06938776
##      CheckingAccountStatus.none CreditHistory.NoCredit.AllPaid
## Bad                   0.1666667                     0.10000000
## Good                  0.4836735                     0.02244898
##      CreditHistory.ThisBank.AllPaid CreditHistory.PaidDuly
## Bad                      0.09523810              0.5571429
## Good                     0.03061224              0.5244898
##      CreditHistory.Delay CreditHistory.Critical Purpose.NewCar
## Bad           0.08571429              0.1619048      0.2952381
## Good          0.07755102              0.3448980      0.2122449
##      Purpose.UsedCar Purpose.Furniture.Equipment Purpose.Radio.Television
## Bad       0.04285714                   0.1904762                0.2142857
## Good      0.10816327                   0.1877551                0.3163265
##      Purpose.DomesticAppliance Purpose.Repairs Purpose.Education
## Bad                 0.01428571      0.01904762        0.07619048
## Good                0.01224490      0.01632653        0.04081633
##      Purpose.Retraining Purpose.Business Purpose.Other
## Bad         0.004761905       0.12380952   0.019047619
## Good        0.010204082       0.08979592   0.006122449
##      SavingsAccountBonds.lt.100 SavingsAccountBonds.100.to.500
## Bad                   0.7238095                      0.1285714
## Good                  0.5489796                      0.1040816
##      SavingsAccountBonds.500.to.1000 SavingsAccountBonds.gt.1000
## Bad                       0.03809524                  0.01904762
## Good                      0.07755102                  0.05918367
##      SavingsAccountBonds.Unknown EmploymentDuration.lt.1
## Bad                   0.09047619               0.2333333
## Good                  0.21020408               0.1367347
##      EmploymentDuration.1.to.4 EmploymentDuration.4.to.7
## Bad                  0.3476190                 0.1285714
## Good                 0.3204082                 0.2163265
##      EmploymentDuration.gt.7 EmploymentDuration.Unemployed
## Bad                0.2142857                    0.07619048
## Good               0.2714286                    0.05510204
##      Personal.Male.Divorced.Seperated Personal.Female.NotSingle
## Bad                        0.04761905                 0.3428571
## Good                       0.04081633                 0.2938776
##      Personal.Male.Single Personal.Male.Married.Widowed
## Bad             0.5380952                    0.07142857
## Good            0.5857143                    0.07959184
##      OtherDebtorsGuarantors.None OtherDebtorsGuarantors.CoApplicant
## Bad                    0.9095238                         0.05238095
## Good                   0.9122449                         0.02244898
##      OtherDebtorsGuarantors.Guarantor Property.RealEstate
## Bad                        0.03809524           0.2000000
## Good                       0.06530612           0.3183673
##      Property.Insurance Property.CarOther Property.Unknown
## Bad           0.2476190         0.3285714        0.2238095
## Good          0.2285714         0.3306122        0.1224490
##      OtherInstallmentPlans.Bank OtherInstallmentPlans.Stores
## Bad                   0.2095238                   0.07142857
## Good                  0.1142857                   0.04081633
```

```
##      OtherInstallmentPlans.None Housing.Rent Housing.Own Housing.ForFree
## Bad                   0.7190476    0.2333333   0.6238095      0.14285714
## Good                  0.8448980    0.1530612   0.7551020      0.09183673
##      Job.UnemployedUnskilled Job.UnskilledResident Job.SkilledEmployee
## Bad               0.02380952             0.1714286           0.6380952
## Good              0.01836735             0.2081633           0.6510204
##      Job.Management.SelfEmp.HighlyQualified
## Bad                               0.1666667
## Good                              0.1224490
##
## Coefficients of linear discriminants:
##                                             LD1
## Duration                          -0.0216856021
## Amount                            -0.0001254021
## InstallmentRatePercentage         -0.1477892602
## ResidenceDuration                  0.0059335640
## Age                                0.0129255354
## NumberExistingCredits             -0.2356809987
## NumberPeopleMaintenance            0.1039756437
## Telephone                         -0.3037740810
## ForeignWorker                     -0.7555334926
## CheckingAccountStatus.lt.0        -0.6802651221
## CheckingAccountStatus.0.to.200    -0.1566746314
## CheckingAccountStatus.gt.200       0.6204429507
## CheckingAccountStatus.none         0.5822805598
## CreditHistory.NoCredit.AllPaid    -0.8002929257
## CreditHistory.ThisBank.AllPaid    -0.8999539610
## CreditHistory.PaidDuly            -0.0910125573
## CreditHistory.Delay                0.2346992535
## CreditHistory.Critical             0.4083883730
## Purpose.NewCar                    -0.5229639366
## Purpose.UsedCar                    0.7796256642
## Purpose.Furniture.Equipment        0.2419758273
## Purpose.Radio.Television           0.1226363293
## Purpose.DomesticAppliance         -0.5208304623
## Purpose.Repairs                   -0.1380512748
## Purpose.Education                 -0.6722419773
## Purpose.Retraining                 0.4130997040
## Purpose.Business                   0.0190668746
## Purpose.Other                      0.6246648314
## SavingsAccountBonds.lt.100        -0.2836749290
## SavingsAccountBonds.100.to.500    -0.0742162810
## SavingsAccountBonds.500.to.1000    0.0751536157
## SavingsAccountBonds.gt.1000        0.4764658400
## SavingsAccountBonds.Unknown        0.3384464680
## EmploymentDuration.lt.1           -0.2307035092
## EmploymentDuration.1.to.4         -0.0926359542
## EmploymentDuration.4.to.7          0.4548014521
## EmploymentDuration.gt.7           -0.0577882211
## EmploymentDuration.Unemployed     -0.1124774496
## Personal.Male.Divorced.Seperated  -0.1276359750
## Personal.Female.NotSingle         -0.0990109104
## Personal.Male.Single               0.0886405260
## Personal.Male.Married.Widowed      0.0652361362
```

```
## OtherDebtorsGuarantors.None            -0.1492637088
## OtherDebtorsGuarantors.CoApplicant     -0.6843004264
## OtherDebtorsGuarantors.Guarantor        0.6096420316
## Property.RealEstate                     0.0609739066
## Property.Insurance                      0.0119492109
## Property.CarOther                       0.1110520535
## Property.Unknown                       -0.3053798731
## OtherInstallmentPlans.Bank             -0.2787415310
## OtherInstallmentPlans.Stores           -0.2033191150
## OtherInstallmentPlans.None              0.2837016837
## Housing.Rent                           -0.2718830803
## Housing.Own                             0.0062887175
## Housing.ForFree                         0.3995714048
## Job.UnemployedUnskilled                -0.1791354529
## Job.UnskilledResident                   0.0737468347
## Job.SkilledEmployee                     0.0089103399
## Job.Management.SelfEmp.HighlyQualified -0.0870595478
```

Plot ROC curves for kNN and LDA on one plot

```
####################################
#### ROC curve for knn
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
knn.pred <- predict(knnFit,test.feature)
confusionMatrix(knn.pred,test.label)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Bad Good
##       Bad   15   12
##       Good  75  198
##
##                Accuracy : 0.71
##                  95% CI : (0.6551, 0.7607)
##     No Information Rate : 0.7
##     P-Value [Acc > NIR] : 0.3793
##
##                   Kappa : 0.1369
##
##  Mcnemar's Test P-Value : 2.989e-11
##
##             Sensitivity : 0.1667
##             Specificity : 0.9429
##          Pos Pred Value : 0.5556
```

```
##          Neg Pred Value : 0.7253
##              Prevalence : 0.3000
##          Detection Rate : 0.0500
##    Detection Prevalence : 0.0900
##       Balanced Accuracy : 0.5548
##
##        'Positive' Class : Bad
##
```

```r
knn.probs <- predict(knnFit,test.feature,type="prob")
head(knn.probs)
```

```
##         Bad      Good
## 1 0.2941176 0.7058824
## 2 0.4705882 0.5294118
## 3 0.1764706 0.8235294
## 4 0.5882353 0.4117647
## 5 0.2352941 0.7647059
## 6 0.6470588 0.3529412
```

```r
knn.ROC <- roc(predictor=knn.probs$Bad,
               response=test.label,
               levels=rev(levels(test.label)))
```

```
## Setting direction: controls < cases
```

```r
knn.ROC$auc
```

```
## Area under the curve: 0.7621
```

```r
plot(knn.ROC,main="ROC curve")
####################################
#### ROC curve for lda
lda.pred <- predict(ldaFit,test.feature)
confusionMatrix(lda.pred,test.label)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Bad Good
##       Bad   40   27
##       Good  50  183
##
##                Accuracy : 0.7433
##                  95% CI : (0.69, 0.7918)
##     No Information Rate : 0.7
##     P-Value [Acc > NIR] : 0.05608
##
##                   Kappa : 0.3408
##
##  Mcnemar's Test P-Value : 0.01217
##
##             Sensitivity : 0.4444
##             Specificity : 0.8714
##          Pos Pred Value : 0.5970
##          Neg Pred Value : 0.7854
##              Prevalence : 0.3000
```

```
##          Detection Rate : 0.1333
##    Detection Prevalence : 0.2233
##       Balanced Accuracy : 0.6579
##
##        'Positive' Class : Bad
##
```

```
lda.probs <- predict(ldaFit,test.feature,type="prob")
head(lda.probs)
```

```
##           Bad       Good
## 2  0.69809154 0.30190846
## 5  0.64721992 0.35278008
## 7  0.05369325 0.94630675
## 11 0.62120103 0.37879897
## 13 0.14188587 0.85811413
## 18 0.95144362 0.04855638
```

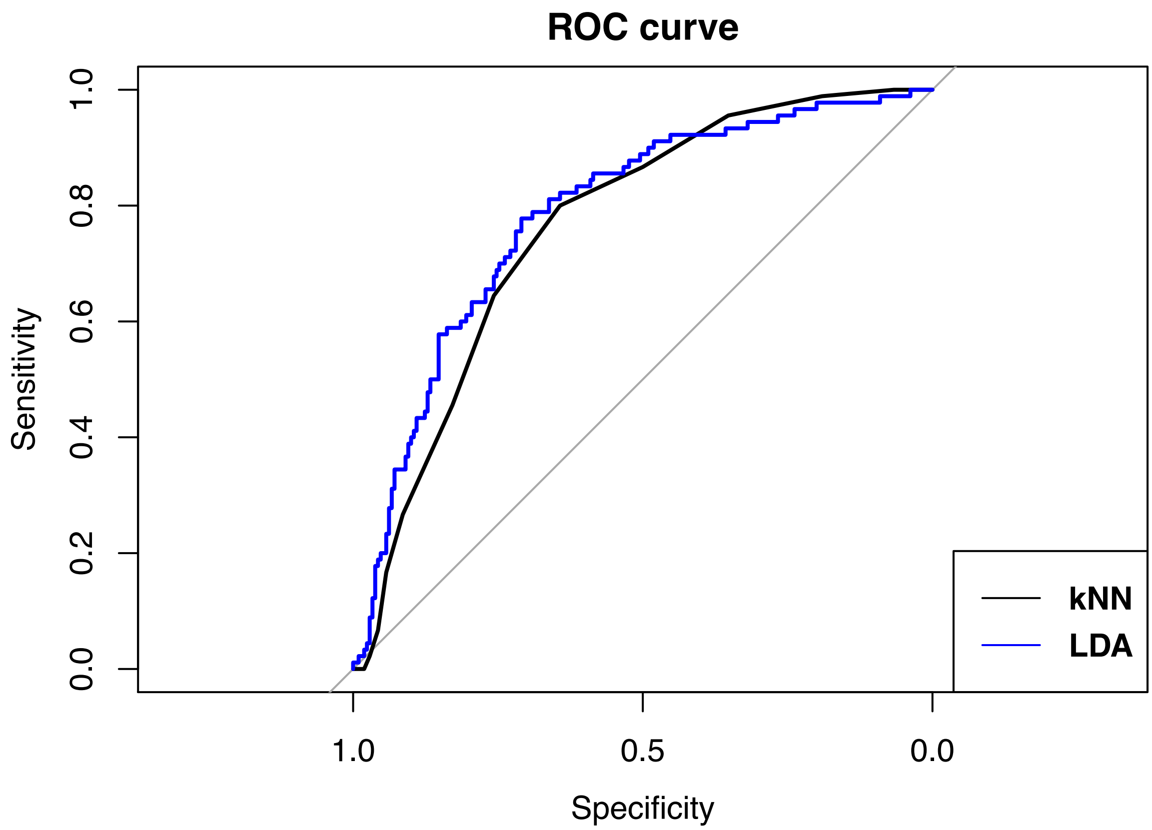
```
lda.ROC <- roc(predictor=lda.probs$Bad,
               response=test.label,
               levels=rev(levels(test.label)))
```

```
## Setting direction: controls < cases
```

```
lda.ROC$auc
```

```
## Area under the curve: 0.7845
```

```
lines(lda.ROC,col="blue") ## add a line to previous plot
legend("bottomright",legend=c("kNN","LDA"),
       col=c("black","blue"),lty=c(1,1),cex=1,text.font=2)
```



Classification on the balanced German Credit data by SMOTE

To use SMOTE to upsampling the minority class, we need to install the `DMwR` package. It's very easy to do this in the `caret` package with the `trainControl` function.

Let's first check the class distribution of the German Credit data.

```
table(GermanCredit$Class)
```

```
##
##  Bad Good
##  300  700
```

To rebalance the dataset, we can simply use the following '`trainControl` setting.

```
####################################
#### knn with smote
#### set up train control
fitControls <- trainControl(## 5-fold CV
  method = "repeatedcv",
  number = 5,
  ## repeated five times
  repeats = 5,
  summaryFunction = twoClassSummary,
  classProbs = TRUE,
  sampling="smote")
#### training process
set.seed(5)
```

```r
knnFits=train(train.feature,train.label, method = "knn",
             trControl = fitControls,
             metric = "ROC",
             preProcess = c("center","scale"),
             tuneLength=10)
```

```
## Loading required package: grid
```

```r
knnFits
```

```
## k-Nearest Neighbors
##
## 700 samples
##  59 predictor
##   2 classes: 'Bad', 'Good'
##
## Pre-processing: centered (59), scaled (59)
## Resampling: Cross-Validated (5 fold, repeated 5 times)
## Summary of sample sizes: 560, 560, 560, 560, 560, 560, ...
## Addtional sampling using SMOTE prior to pre-processing
##
## Resampling results across tuning parameters:
##
##   k   ROC        Sens       Spec
##    5  0.6620554  0.5647619  0.6693878
##    7  0.6698154  0.5828571  0.6624490
##    9  0.6756803  0.5942857  0.6591837
##   11  0.6849563  0.6142857  0.6600000
##   13  0.6989796  0.6266667  0.6612245
##   15  0.7036443  0.6266667  0.6657143
##   17  0.7095967  0.6542857  0.6436735
##   19  0.7077162  0.6580952  0.6481633
##   21  0.7191059  0.6666667  0.6481633
##   23  0.7217736  0.6695238  0.6432653
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was k = 23.
```

The same applies to LDA.

```r
####################################
#### lda with SMOTE
set.seed(5)
ldaFits=train(train.feature,train.label, method = "lda",
             trControl = trainControl(sampling="smote"))
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear
## Warning: model fit failed for Resample21: parameter=none Error in lda.default(x, grouping, ...) :
##   variable 26 appears to be constant within groups
## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear

## Warning in lda.default(x, grouping, ...): variables are collinear
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info =
## trainInfo, : There were missing values in resampled performance measures.
## Warning in lda.default(x, grouping, ...): variables are collinear
```

```
ldaFits
```

```
## Linear Discriminant Analysis
##
## 700 samples
##  59 predictor
##   2 classes: 'Bad', 'Good'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 700, 700, 700, 700, 700, 700, ...
```

```
## Addtional sampling using SMOTE
##
## Resampling results:
##
##   Accuracy   Kappa
##   0.7286711  0.3651564
```

```
ldaFits$finalModel
```

```
## Call:
## lda(x, y)
##
## Prior probabilities of groups:
##       Bad      Good
## 0.4285714 0.5714286
##
## Group means:
##      Duration   Amount InstallmentRatePercentage ResidenceDuration
## Bad  24.74778 3918.292                  3.140612          2.835548
## Good 18.75833 2993.985                  2.928571          2.936905
##           Age NumberExistingCredits NumberPeopleMaintenance Telephone
## Bad  33.04800              1.323518                1.141985 0.6657055
## Good 36.80119              1.440476                1.173810 0.5821429
##      ForeignWorker CheckingAccountStatus.lt.0
## Bad      0.9874730                  0.4842312
## Good     0.9333333                  0.2011905
##      CheckingAccountStatus.0.to.200 CheckingAccountStatus.gt.200
## Bad                       0.3415638                   0.02598997
## Good                      0.2321429                   0.05714286
##      CheckingAccountStatus.none CreditHistory.NoCredit.AllPaid
## Bad                   0.1482151                     0.08553256
## Good                  0.5095238                     0.02380952
##      CreditHistory.ThisBank.AllPaid CreditHistory.PaidDuly
## Bad                      0.08635888              0.6140479
## Good                     0.02976190              0.5083333
##      CreditHistory.Delay CreditHistory.Critical Purpose.NewCar
## Bad           0.07280502              0.1412556      0.2900816
## Good          0.07380952              0.3642857      0.2154762
##      Purpose.UsedCar Purpose.Furniture.Equipment Purpose.Radio.Television
## Bad       0.03796406                   0.2149873                0.2132091
## Good      0.12500000                   0.1964286                0.2904762
##      Purpose.DomesticAppliance Purpose.Repairs Purpose.Education
## Bad                0.009749694      0.01897371        0.07472765
## Good               0.009523810      0.01309524        0.04166667
##      Purpose.Retraining Purpose.Business Purpose.Other
## Bad         0.004113046       0.12189297   0.014300808
## Good        0.009523810       0.09166667   0.007142857
##      SavingsAccountBonds.lt.100 SavingsAccountBonds.100.to.500
## Bad                   0.7588172                      0.1141553
## Good                  0.5178571                      0.1202381
##      SavingsAccountBonds.500.to.1000 SavingsAccountBonds.gt.1000
## Bad                       0.02383926                  0.01615081
## Good                      0.09523810                  0.05714286
##      SavingsAccountBonds.Unknown EmploymentDuration.lt.1
## Bad                    0.08703749               0.2210477
```

```
## Good                 0.20952381              0.1226190
##      EmploymentDuration.1.to.4 EmploymentDuration.4.to.7
## Bad                 0.3667442                 0.1276074
## Good                0.2928571                 0.2285714
##      EmploymentDuration.gt.7 EmploymentDuration.Unemployed
## Bad               0.2161961                    0.06840455
## Good              0.3035714                    0.05238095
##      Personal.Male.Divorced.Seperated Personal.Female.NotSingle
## Bad                        0.03946275                 0.3477845
## Good                       0.04047619                 0.2726190
##      Personal.Male.Single Personal.Male.Married.Widowed
## Bad             0.5540571                    0.05869565
## Good            0.6297619                    0.05714286
##      OtherDebtorsGuarantors.None OtherDebtorsGuarantors.CoApplicant
## Bad                    0.9257508                         0.04381567
## Good                   0.9035714                         0.03095238
##      OtherDebtorsGuarantors.Guarantor Property.RealEstate
## Bad                        0.03043352           0.2036505
## Good                       0.06547619           0.3333333
##      Property.Insurance Property.CarOther Property.Unknown
## Bad           0.2345146         0.3533878        0.2084471
## Good          0.2309524         0.3190476        0.1166667
##      OtherInstallmentPlans.Bank OtherInstallmentPlans.Stores
## Bad                   0.1840532                   0.06224644
## Good                  0.1357143                   0.03809524
##      OtherInstallmentPlans.None Housing.Rent Housing.Own Housing.ForFree
## Bad                   0.7537004    0.2227055   0.6321113      0.14518322
## Good                  0.8261905    0.1523810   0.7702381      0.07738095
##      Job.UnemployedUnskilled Job.UnskilledResident Job.SkilledEmployee
## Bad               0.02092185             0.1650926           0.6627085
## Good              0.01904762             0.2202381           0.6476190
##      Job.Management.SelfEmp.HighlyQualified
## Bad                               0.1512771
## Good                              0.1130952
##
## Coefficients of linear discriminants:
##                                            LD1
## Duration                         -0.0244286784
## Amount                           -0.0001112974
## InstallmentRatePercentage        -0.1774944223
## ResidenceDuration                 0.1139035394
## Age                               0.0131778233
## NumberExistingCredits            -0.2166177429
## NumberPeopleMaintenance           0.0955659326
## Telephone                        -0.4395926361
## ForeignWorker                    -1.3272636675
## CheckingAccountStatus.lt.0       -0.5738128558
## CheckingAccountStatus.0.to.200   -0.2732903344
## CheckingAccountStatus.gt.200      0.5927366592
## CheckingAccountStatus.none        0.7121135553
## CreditHistory.NoCredit.AllPaid   -0.5496605583
## CreditHistory.ThisBank.AllPaid   -1.0287060991
## CreditHistory.PaidDuly           -0.1647604209
## CreditHistory.Delay               0.2310367405
```

```
## CreditHistory.Critical                     0.5276607394
## Purpose.NewCar                            -0.5130789681
## Purpose.UsedCar                            1.0370197740
## Purpose.Furniture.Equipment                0.1821540900
## Purpose.Radio.Television                  -0.0496879922
## Purpose.DomesticAppliance                 -0.5696345452
## Purpose.Repairs                           -0.2355370012
## Purpose.Education                         -0.4598790130
## Purpose.Retraining                         0.6495264970
## Purpose.Business                           0.0932185827
## Purpose.Other                              1.0170482818
## SavingsAccountBonds.lt.100                -0.4166882259
## SavingsAccountBonds.100.to.500             0.1948563753
## SavingsAccountBonds.500.to.1000            0.5434037684
## SavingsAccountBonds.gt.1000                0.2119387166
## SavingsAccountBonds.Unknown                0.2439059957
## EmploymentDuration.lt.1                   -0.0639323549
## EmploymentDuration.1.to.4                 -0.2363971773
## EmploymentDuration.4.to.7                  0.5205496303
## EmploymentDuration.gt.7                   -0.0754261085
## EmploymentDuration.Unemployed             -0.0477764393
## Personal.Male.Divorced.Seperated           0.0727422561
## Personal.Female.NotSingle                 -0.1862317245
## Personal.Male.Single                       0.1216538077
## Personal.Male.Married.Widowed              0.1322873733
## OtherDebtorsGuarantors.None               -0.2177604267
## OtherDebtorsGuarantors.CoApplicant        -0.5081242625
## OtherDebtorsGuarantors.Guarantor           0.7143743147
## Property.RealEstate                        0.0765268182
## Property.Insurance                        -0.0129252896
## Property.CarOther                          0.0117097857
## Property.Unknown                          -0.1190983567
## OtherInstallmentPlans.Bank                -0.1710244604
## OtherInstallmentPlans.Stores              -0.1203050321
## OtherInstallmentPlans.None                 0.1724743962
## Housing.Rent                              -0.1555102151
## Housing.Own                                0.1286621217
## Housing.ForFree                           -0.0318793492
## Job.UnemployedUnskilled                   -0.2148957897
## Job.UnskilledResident                      0.1408840763
## Job.SkilledEmployee                        0.0596664625
## Job.Management.SelfEmp.HighlyQualified -0.2843053060
```

Now we can draw the ROC curves the same as before.

```r
####################################
#### ROC curve for knn with SMOTE
knn.preds <- predict(knnFits,test.feature)
confusionMatrix(knn.preds,test.label)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Bad Good
##       Bad   68   76
```

```
##       Good   22  134
##
##                Accuracy : 0.6733
##                  95% CI : (0.6171, 0.7261)
##     No Information Rate : 0.7
##     P-Value [Acc > NIR] : 0.8577
##
##                   Kappa : 0.336
##
##  Mcnemar's Test P-Value : 8.612e-08
##
##             Sensitivity : 0.7556
##             Specificity : 0.6381
##          Pos Pred Value : 0.4722
##          Neg Pred Value : 0.8590
##              Prevalence : 0.3000
##          Detection Rate : 0.2267
##    Detection Prevalence : 0.4800
##       Balanced Accuracy : 0.6968
##
##        'Positive' Class : Bad
##
```

```
knn.probss <- predict(knnFits,test.feature,type="prob")
head(knn.probss)
```

```
##         Bad       Good
## 1 0.5833333 0.41666667
## 2 0.9565217 0.04347826
## 3 0.3478261 0.65217391
## 4 0.4800000 0.52000000
## 5 0.3333333 0.66666667
## 6 0.7826087 0.21739130
```

```
knn.ROCs <- roc(predictor=knn.probss$Bad,
               response=test.label,
               levels=rev(levels(test.label)))
```

```
## Setting direction: controls < cases
```

```
knn.ROCs$auc
```

```
## Area under the curve: 0.7577
```

```
plot(knn.ROCs,main="ROC curve")
####################################
#### ROC curve for lda
lda.preds <- predict(ldaFits,test.feature)
confusionMatrix(lda.preds,test.label)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Bad Good
##       Bad   61   54
##       Good  29  156
##
```

```
##                Accuracy : 0.7233
##                  95% CI : (0.669, 0.7732)
##     No Information Rate : 0.7
##     P-Value [Acc > NIR] : 0.20722
##
##                   Kappa : 0.3897
##
##  Mcnemar's Test P-Value : 0.00843
##
##             Sensitivity : 0.6778
##             Specificity : 0.7429
##          Pos Pred Value : 0.5304
##          Neg Pred Value : 0.8432
##              Prevalence : 0.3000
##          Detection Rate : 0.2033
##    Detection Prevalence : 0.3833
##       Balanced Accuracy : 0.7103
##
##        'Positive' Class : Bad
##
```

```
lda.probss <- predict(ldaFits,test.feature,type="prob")
head(lda.probss)
```

```
##           Bad       Good
## 2  0.95622771 0.04377229
## 5  0.88488545 0.11511455
## 7  0.02978536 0.97021464
## 11 0.90401455 0.09598545
## 13 0.52015674 0.47984326
## 18 0.94757092 0.05242908
```

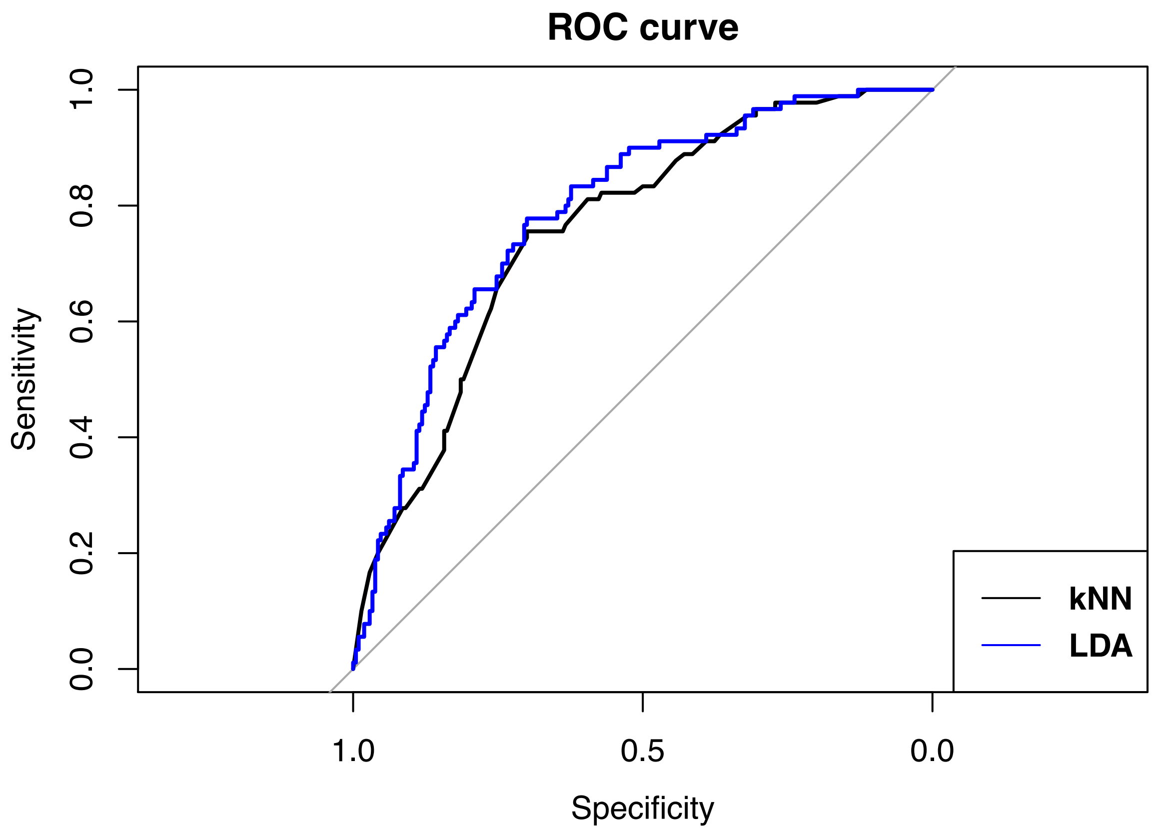
```
lda.ROCs <- roc(predictor=lda.probss$Bad,
               response=test.label,
               levels=rev(levels(test.label)))
```

```
## Setting direction: controls < cases
```

```
lda.ROCs$auc
```

```
## Area under the curve: 0.7878
```

```
lines(lda.ROCs,col="blue")
legend("bottomright",legend=c("kNN","LDA"),
       col=c("black","blue"),lty=c(1,1),cex=1,text.font=2)
```


ROC curve
Sensitivity
0.0
0.2
0.4
0.6
0.8
1.0
Specificity
1.0
0.5
0.0
kNN
LDA