Problem Set2_Yile Chen

Yile Chen

2/9/2021

```r
library(tidyverse)
library(here) # for loading data; this is *optional*
library(tidymodels) # for accuracy, splitting, etc.
library(class) # for knn()
library(MASS) # for lda() and qda()
library(caret)
library(pROC)
library(MLmetrics)
```

Question 1

```r
sim <- function(n)
  #set up the dataset
  {data <- tibble(x1 = runif(n, -1, 1), x2 = runif(n, -1, 1),
                  y = x1 + x1^2 + x2 + x2^2 + rnorm(n, 0, 1), #plus error term
                  y_class = y >= 0)

  #split
  split <- initial_split(data, prop = 0.8)
  train <- training(split)
  test <- testing(split)

  # fit the LDA and QDA classifier
  lda_mod <- lda(y_class ~ x1 + x1^2 + x2 + x2^2, data = train)
  qda_mod <- qda(y_class ~ x1 + x1^2 + x2 + x2^2, data = train)

  #get error rates
  lda_train <- mean(predict(lda_mod, train)$class!=train$y_class)
  lda_test <- mean(predict(lda_mod, test)$class!=test$y_class)
  qda_train <- mean(predict(qda_mod, train)$class!=train$y_class)
  qda_test <- mean(predict(qda_mod, test)$class!=test$y_class)

  return(tibble(lda_train,  lda_test, qda_train, qda_test))}

#simulate 1000 times
sim_result <- data.frame()
for (i in 1:1000){sim_result <- rbind(sim_result, sim(1000))}

#get the numerical result
skimr::skim(sim_result)
```

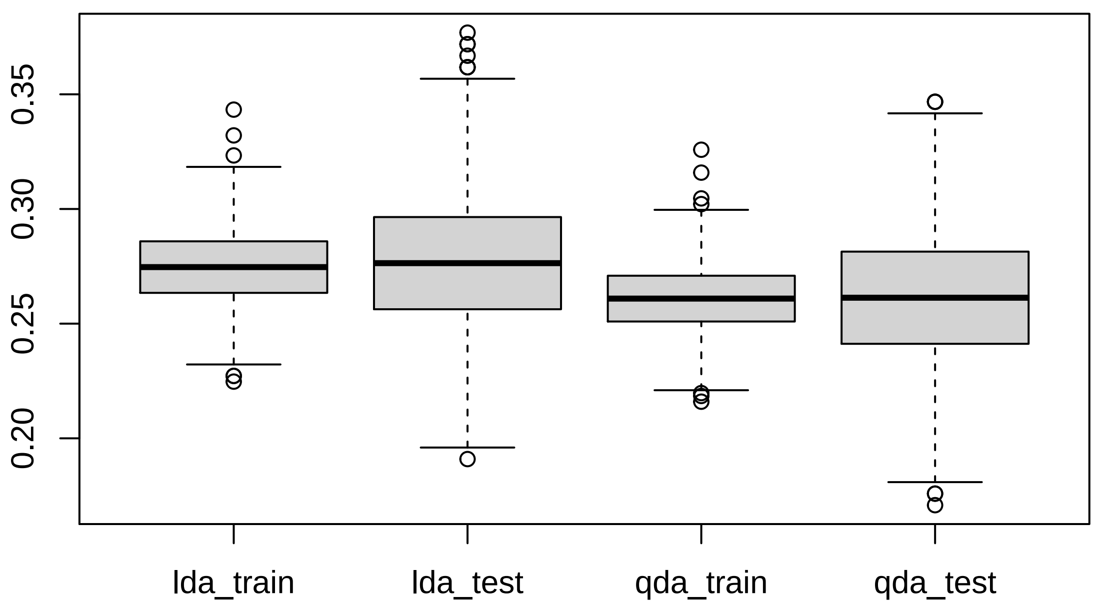
Table 1: Data summary

Name	sim_result
Number of rows	1000
Number of columns	4
Column type frequency:	
numeric	4
Group variables	None

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
lda_train	0	1	0.27	0.02	0.22	0.26	0.27	0.29	0.34	
lda_test	0	1	0.28	0.03	0.19	0.26	0.28	0.30	0.38	
qda_train	0	1	0.26	0.02	0.22	0.25	0.26	0.27	0.33	
qda_test	0	1	0.26	0.03	0.17	0.24	0.26	0.28	0.35	

```r
#get the visualization
boxplot(sim_result)
```



As shown, for both methods, testing error is larger than training error and is more dispersed. QDA has lower training error and testing error than LDA. Therefore, the above analysis presents evidence that when the Bayes decision boundary is non-linear, QDA outperforms LDA regarding both on the training set and the test set.

The reason accounting for QDA's better performance are: QDA assumes a quadratic decision boundary, and can thus capture the non-linear relationship between y and x1 & x2 more accurately. The flexibility of QDA leads to lower bias without sacrificing too much on variance - shown by the simulation error rates.

Question 2

For this question, I discussed and collaberated with Yier Ling, and shared thoughts with Jingfei Zhu, Boya Fu, Xi Cheng.

```
#load and preprocess data#
anes <- read_csv("anes_pilot_2016.csv")

anes_short <- anes %>%
  dplyr::select(pid3, ideo5, fttrump, ftobama, fthrc, ftrubio) %>%
  mutate(democrat = as.factor(ifelse(pid3 == 1, 1, 0)),
         fttrump = replace(fttrump, fttrump > 100, NA),
         ftobama = replace(ftobama, ftobama > 100, NA),
         fthrc = replace(fthrc, fthrc > 100, NA),
         ftrubio = replace(ftrubio, ftrubio > 100, NA))%>%
  drop_na()

anes_short <- anes_short %>%
  dplyr::select(-c(pid3)) %>%
  relocate(c(democrat)) #%>%

levels(anes_short$democrat) <- (c("others", "democrat"))

skimr::skim(anes_short)
```

Table 3: Data summary

Name	anes_short
Number of rows	1182
Number of columns	6
Column type frequency:	
factor	1
numeric	5
Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
democrat	0	1	FALSE	2	oth: 731, dem: 451

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
ideo5	0	1	3.23	1.37	1	2	3	4.00	6	
fttrump	0	1	38.21	36.52	0	2	30	72.00	100	
ftobama	0	1	48.58	38.07	0	5	53	87.00	100	
fthrc	0	1	43.10	36.49	0	3	46	76.00	100	
ftrubio	0	1	41.58	28.06	0	15	47	60.75	100	

```r
#Split#
set.seed(1234)
split_anes <- initial_split(anes_short, prop = 0.8)
train_anes <- training(split_anes)
test_anes <- testing(split_anes)
```

```r
#Logit#
set.seed(1234)
logit_fit <- train(democrat ~., train_anes,
                  method = "glm", family = "binomial",
                  trControl = trainControl(method = "cv", number = 10,
                                           returnData = TRUE, savePredictions = TRUE,
                                           classProbs = TRUE, summaryFunction = multiClassSummary))

logit_ROC <- roc(logit_fit$pred$obs, logit_fit$pred$democrat, auc = TRUE, plot = FALSE, print.auc=TRUE)
logit_AUC <- logit_ROC$auc

logit_error <- 1-logit_fit$results$Accuracy
logit_error
```

```
## [1] 0.2029787
```

```r
#LDA#
set.seed(1234)
LDA_fit <- train(democrat ~., train_anes,
                  method = "lda",
                  trControl = trainControl(method = "cv", number = 10,
                                           returnData = TRUE, savePredictions = TRUE,
                                           classProbs = TRUE, summaryFunction = multiClassSummary))

LDA_ROC <- roc(LDA_fit$pred$obs, LDA_fit$pred$democrat, auc = TRUE, plot = FALSE, print.auc=TRUE)

LDA_AUC <- LDA_ROC$auc

LDA_error <- 1-LDA_fit$results$Accuracy
LDA_error
```

```
## [1] 0.2008511
```

```r
#QDA#
set.seed(1234)
QDA_fit <- train(democrat ~., train_anes,
                  method = "qda",
                  trControl = trainControl(method = "cv", number = 10,
                                           returnData = TRUE, savePredictions = TRUE,
                                           classProbs = TRUE, summaryFunction = multiClassSummary))

QDA_ROC <- roc(QDA_fit$pred$obs, QDA_fit$pred$democrat, auc = TRUE, plot = FALSE, print.auc=TRUE)

QDA_AUC <- QDA_ROC$auc

QDA_error <- 1-QDA_fit$results$Accuracy
QDA_error
```

```
## [1] 0.2029227
```

```r
set.seed(1234)
#1NN#
knn1_fit <- train(democrat ~., train_anes,
                  method = "knn",
                  tuneGrid = expand.grid(k = 1),
                  trControl = trainControl(method = "cv", number = 10,
                                           returnData = TRUE, savePredictions = TRUE,
                                           classProbs = TRUE, summaryFunction = multiClassSummary))
knn1_ROC <- roc(knn1_fit$pred$obs, knn1_fit$pred$democrat, auc = TRUE, plot = FALSE, print.auc=TRUE)

knn1_AUC <- knn1_ROC$auc
knn1_error <- 1-knn1_fit$results$Accuracy
knn1_error
```

```
## [1] 0.2513998
```

```r
#2NN#
knn2_fit <- train(democrat ~., train_anes,
                  method = "knn",
                  tuneGrid = expand.grid(k = 2),
                  trControl = trainControl(method = "cv", number = 10,
                                           returnData = TRUE, savePredictions = TRUE,
                                           classProbs = TRUE, summaryFunction = multiClassSummary))
knn2_ROC <- roc(knn2_fit$pred$obs, knn2_fit$pred$democrat, auc = TRUE, plot = FALSE, print.auc=TRUE)

knn2_AUC <- knn2_ROC$auc
knn2_error <- 1-knn2_fit$results$Accuracy
knn2_error
```

```
## [1] 0.2578499
```

```r
#3NN#
knn3_fit <- train(democrat ~., train_anes,
                  method = "knn",
                  tuneGrid = expand.grid(k = 3),
                  trControl = trainControl(method = "cv", number = 10,
                                           returnData = TRUE, savePredictions = TRUE,
                                           classProbs = TRUE, summaryFunction = multiClassSummary))
knn3_ROC <- roc(knn3_fit$pred$obs, knn3_fit$pred$democrat, auc = TRUE, plot = FALSE, print.auc=TRUE)

knn3_AUC <- knn3_ROC$auc
knn3_error <- 1-knn3_fit$results$Accuracy
knn3_error
```

```
## [1] 0.227234
```

```r
#4NN#
knn4_fit <- train(democrat ~., train_anes,
                  method = "knn",
                  tuneGrid = expand.grid(k = 4),
                  trControl = trainControl(method = "cv", number = 10,
                                           returnData = TRUE, savePredictions = TRUE,
                                           classProbs = TRUE, summaryFunction = multiClassSummary))
knn4_ROC <- roc(knn4_fit$pred$obs, knn4_fit$pred$democrat, auc = TRUE, plot = FALSE, print.auc=TRUE)

knn4_AUC <- knn4_ROC$auc
```

```r
knn4_error <- 1-knn4_fit$results$Accuracy
knn4_error
```

```
## [1] 0.2325756
```

```r
#5NN#
knn5_fit <- train(democrat ~., train_anes,
                  method = "knn",
                  tuneGrid = expand.grid(k = 5),
                  trControl = trainControl(method = "cv", number = 10,
                                           returnData = TRUE, savePredictions = TRUE,
                                           classProbs = TRUE, summaryFunction = multiClassSummary))
knn5_ROC <- roc(knn5_fit$pred$obs, knn5_fit$pred$democrat, auc = TRUE, plot = FALSE, print.auc=TRUE)

knn5_AUC <- knn5_ROC$auc
knn5_error <- 1-knn5_fit$results$Accuracy
knn5_error
```

```
## [1] 0.2166853
```

```r
#6NN#
knn6_fit <- train(democrat ~., train_anes,
                  method = "knn",
                  tuneGrid = expand.grid(k = 6),
                  trControl = trainControl(method = "cv", number = 10,
                                           returnData = TRUE, savePredictions = TRUE,
                                           classProbs = TRUE, summaryFunction = multiClassSummary))
knn6_ROC <- roc(knn6_fit$pred$obs, knn6_fit$pred$democrat, auc = TRUE, plot = FALSE, print.auc=TRUE)

knn6_AUC <- knn6_ROC$auc
knn6_error <- 1-knn6_fit$results$Accuracy
knn6_error
```

```
## [1] 0.212486
```

```r
#7NN#
knn7_fit <- train(democrat ~., train_anes,
                  method = "knn",
                  tuneGrid = expand.grid(k = 7),
                  trControl = trainControl(method = "cv", number = 10,
                                           returnData = TRUE, savePredictions = TRUE,
                                           classProbs = TRUE, summaryFunction = multiClassSummary))
knn7_ROC <- roc(knn7_fit$pred$obs, knn7_fit$pred$democrat, auc = TRUE, plot = FALSE, print.auc=TRUE)

knn7_AUC <- knn7_ROC$auc
knn7_error <- 1-knn7_fit$results$Accuracy
knn7_error
```

```
## [1] 0.2166517
```

```r
#8NN#
knn8_fit <- train(democrat ~., train_anes,
                  method = "knn",
                  tuneGrid = expand.grid(k = 8),
                  trControl = trainControl(method = "cv", number = 10,
                                           returnData = TRUE, savePredictions = TRUE,
                                           classProbs = TRUE, summaryFunction = multiClassSummary))
```

```r
knn8_ROC <- roc(knn8_fit$pred$obs, knn8_fit$pred$democrat, auc = TRUE, plot = FALSE, print.auc=TRUE)

knn8_AUC <- knn8_ROC$auc
knn8_error <- 1-knn8_fit$results$Accuracy
knn8_error
```

```
## [1] 0.2145577
```

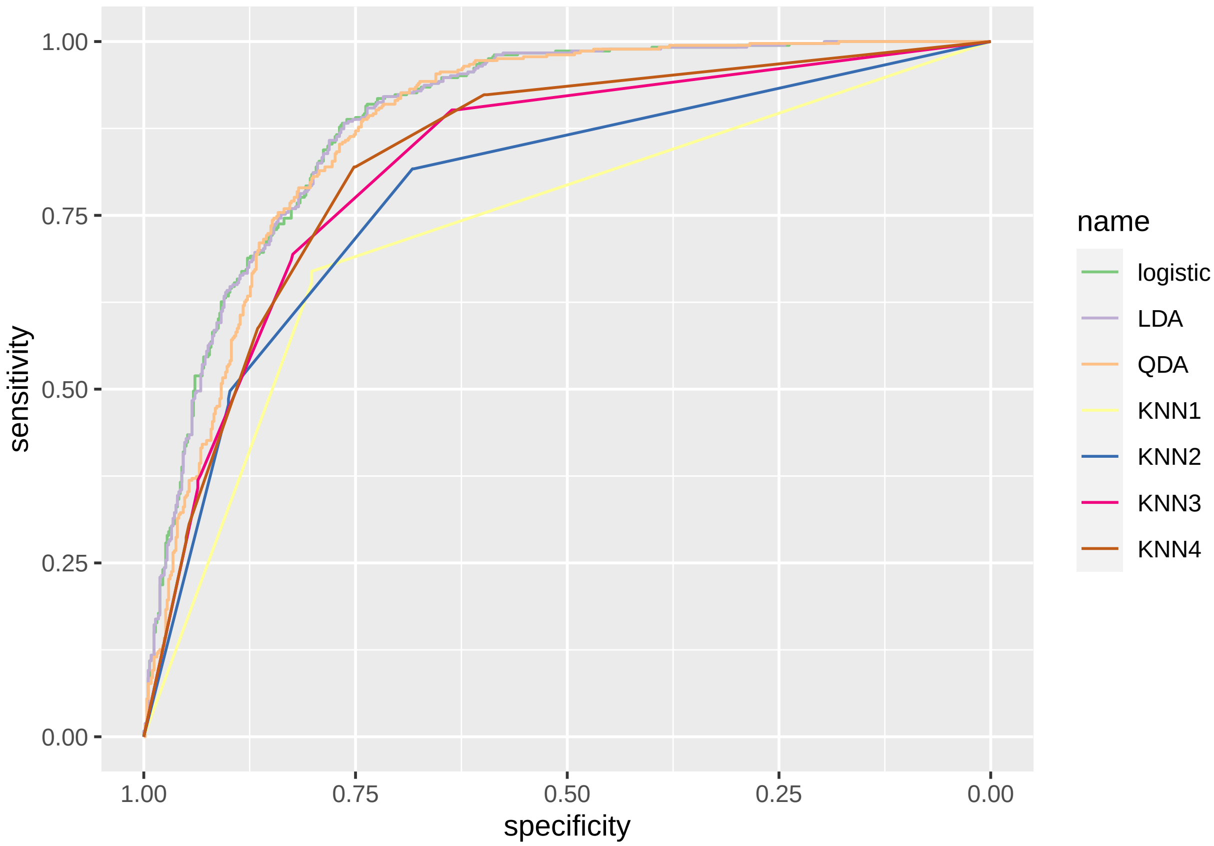
```r
#9NN#
knn9_fit <- train(democrat ~., train_anes,
                 method = "knn",
                 tuneGrid = expand.grid(k = 9),
                 trControl = trainControl(method = "cv", number = 10,
                                          returnData = TRUE, savePredictions = TRUE,
                                          classProbs = TRUE, summaryFunction = multiClassSummary))
knn9_ROC <- roc(knn9_fit$pred$obs, knn9_fit$pred$democrat, auc = TRUE, plot = FALSE, print.auc=TRUE)

knn9_AUC <- knn9_ROC$auc
knn9_error <- 1-knn9_fit$results$Accuracy
knn9_error
```

```
## [1] 0.206159
```

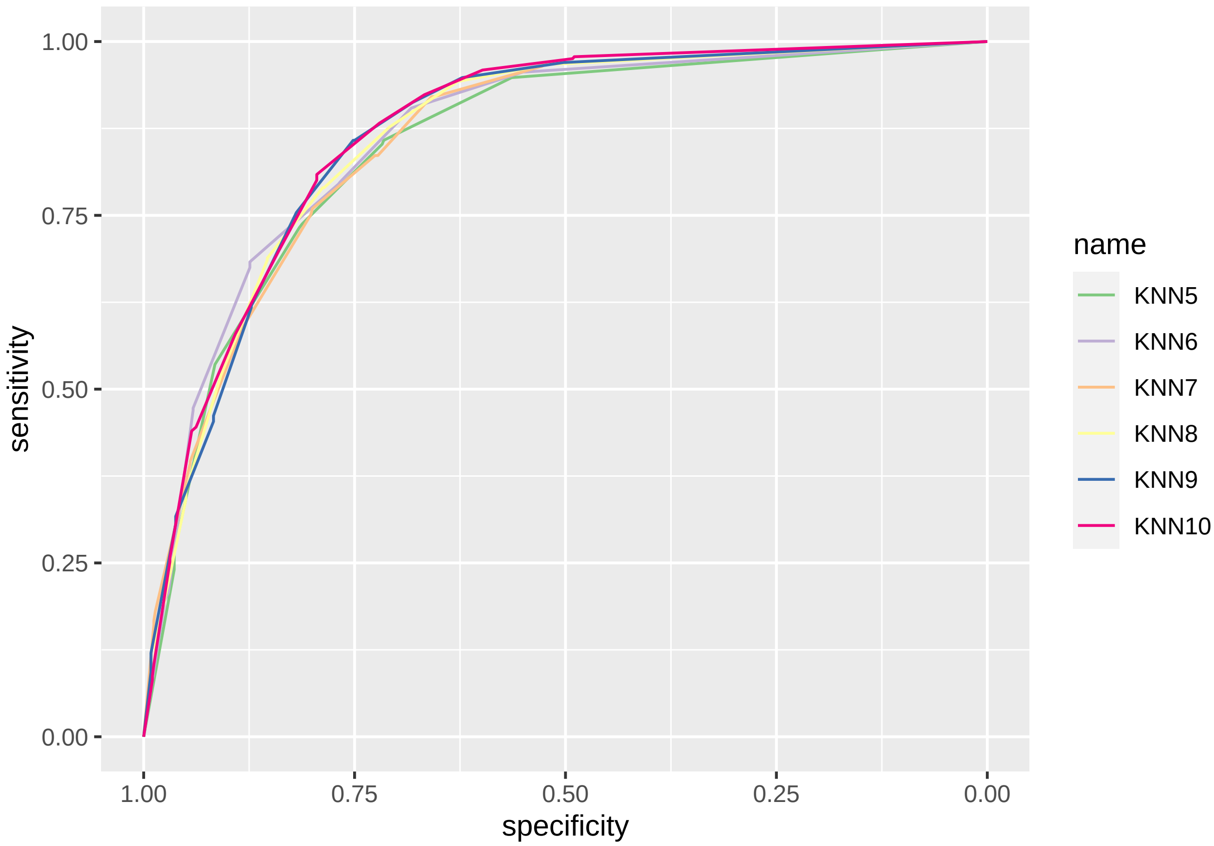
```r
#10NN#
knn10_fit <- train(democrat ~., train_anes,
                 method = "knn",
                 tuneGrid = expand.grid(k = 10),
                 trControl = trainControl(method = "cv", number = 10,
                                          returnData = TRUE, savePredictions = TRUE,
                                          classProbs = TRUE, summaryFunction = multiClassSummary))
knn10_ROC <- roc(knn10_fit$pred$obs, knn10_fit$pred$democrat, auc = TRUE, plot = FALSE, print.auc=TRUE)

knn10_AUC <- knn10_ROC$auc
knn10_error <- 1-knn10_fit$results$Accuracy
knn10_error
```

```
## [1] 0.2061478
```

```r
#Demostrate results numerically and visually
par(mfrow = c(1,2))
ggroc(list(logit_ROC,LDA_ROC,QDA_ROC,knn1_ROC,knn2_ROC,knn3_ROC,knn4_ROC)) +
   scale_color_brewer(palette = "Accent", labels = c("logistic","LDA","QDA","KNN1","KNN2","KNN3","KNN4"
```




```
ggroc(list(knn5_ROC,knn6_ROC,knn7_ROC,knn8_ROC,knn9_ROC,knn10_ROC)) +
  scale_color_brewer(palette = "Accent", labels = c("KNN5","KNN6","KNN7","KNN8","KNN9","KNN10"))
```




```r
erroc_table <- tibble(logit_error,LDA_error,QDA_error,knn1_error,knn2_error,knn3_error,knn4_error,knn5_e
t(erroc_table)
```

```
##                   [,1]
## logit_error 0.2029787
## LDA_error   0.2008511
## QDA_error   0.2029227
## knn1_error  0.2513998
## knn2_error  0.2578499
## knn3_error  0.2272340
## knn4_error  0.2325756
## knn5_error  0.2166853
## knn6_error  0.2124860
## knn7_error  0.2166517
## knn8_error  0.2145577
## knn9_error  0.2061590
## knn10_error 0.2061478
```

```r
auc_table <- tibble(logit_AUC,LDA_AUC,QDA_AUC,knn1_AUC,knn2_AUC,knn3_AUC,knn4_AUC,knn5_AUC,knn6_AUC,knn7
t(auc_table)
```

```
##                 [,1]
## logit_AUC 0.8891841
## LDA_AUC   0.8888119
## QDA_AUC   0.8784200
## knn1_AUC  0.7339363
## knn2_AUC  0.7864801
## knn3_AUC  0.8252968
```

```
## knn4_AUC  0.8331849
## knn5_AUC  0.8526357
## knn6_AUC  0.8642430
## knn7_AUC  0.8597395
## knn8_AUC  0.8630559
## knn9_AUC  0.8660001
## knn10_AUC 0.8710453
```

ROC curves demostrate the relationship between specificity and sensitivity, that is, it plots the true positive rate (TPR) against the false positive rate (FPR). AUC stands for area under the ROC curve, ranging from 0 to 1, with 0 means the model's predictions are 100% wrong and 1 means the predictions are 100% correct. AUC equals to the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one

According to the above results, we can see that the performace of logistic regression, LDA and QDA are the pretty close to each other and are better than the others. In terms of classification error rate, LDA outperform the rest and is closely followed by logistic regression annd QDA. Regarding AUC, logistic regression and LDA ranked top; and seen from the ROC, the curves of logistic, LDA and QDA models are closer to the upper left corner than the rest of KNN curves. In general, I think LDA performs performs slightly better than the other two parametric models, and much better than the non-parametric models. Therefore, I choose LDA as the best model for question2.f.

As logistic regression and LDA methods are closely connected, it is not surprising to see that the two perform rather similarly. The fact that QDA performs better than KNN in general further indicates the importance of introducing assumptions on the data distributions with limited data. In addition, the fact that QDA does not perform vastly differently to LDA may indicate that there is no need to include quadratic terms under this circumstance. Lastly, how the KNN performances changes with the values of k validates the importance of choosing the right k in using KNN methods in general.

```r
#trian the best model - lda - on the whole training set
set.seed(1234)
lda_train_anes <- lda(democrat ~ .,
               data = train_anes)
lda_train_anes
```

```
## Call:
## lda(democrat ~ ., data = train_anes)
##
## Prior probabilities of groups:
##    others  democrat
## 0.6131078 0.3868922
##
## Group means:
##             ideo5  fttrump  ftobama    fthrc  ftrubio
## others   3.705172 50.63448 29.80000 25.52241 48.96724
## democrat 2.494536 17.74863 78.12295 71.44262 29.45628
##
## Coefficients of linear discriminants:
##                  LD1
## ideo5   -0.143865158
## fttrump -0.005209932
## ftobama  0.013150809
## fthrc    0.016821913
## ftrubio -0.004610503
```

```r
#predict
lda_test_pred_anes <- predict(lda_train_anes, test_anes)
```

```r
#calculate the test error rate
lda_test_error_anes <- mean(lda_test_pred_anes$class != test_anes$democrat)
lda_test_error_anes
```

```
## [1] 0.1822034
```

```r
#confusion matrix
lda_cm_anes <- table(predicted=lda_test_pred_anes$class, true = test_anes$democrat)
lda_cm_anes
```

```
##           true
## predicted  others democrat
##   others      121       13
##   democrat     30       72
```

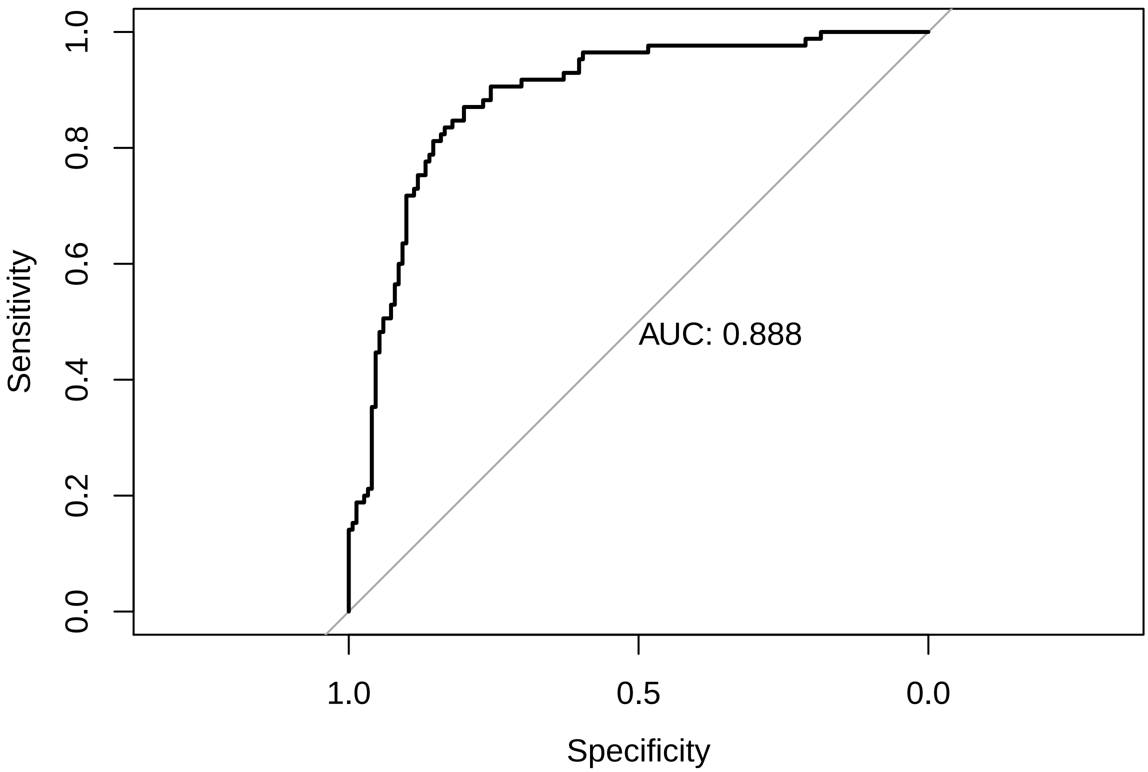
```r
lda_cm_prop_anes <- lda_cm_anes %>% prop.table() %>% round(6)
lda_cm_prop_anes
```

```
##           true
## predicted    others democrat
##   others   0.512712 0.055085
##   democrat 0.127119 0.305085
```

```r
overall_error <- lda_test_error_anes
false_positive <- lda_cm_anes[2,1]/sum(lda_cm_anes[,1])
true_positive <- lda_cm_anes[2,2]/sum(lda_cm_anes[,2])
type_I <- false_positive
type_II <- 1-true_positive
#present the error statistics
tibble(overall_error, false_positive, true_positive, type_I, type_II)
```

```
## # A tibble: 1 x 5
##   overall_error false_positive true_positive type_I type_II
##           <dbl>          <dbl>         <dbl>  <dbl>   <dbl>
## 1         0.182          0.199         0.847  0.199   0.153
```

```r
#present the metrics numerically and visually
lda_ROC <- roc(test_anes$democrat,lda_test_pred_anes$posterior[,"democrat"], plot = TRUE, print.auc = TI
```




```r
tibble(lda_test_error_anes, lda_auc = lda_ROC$auc)
```

```
## # A tibble: 1 x 2
##   lda_test_error_anes lda_auc
##                 <dbl> <auc>
## 1               0.182 0.8881963
```

The test error rate here (0.1822) is lower than LDA's average error with cross validation, which is reasonable because the randomness of spliting. The low overall error along with low false positive rate(0.1986 - typeI error) and typeII error suggest good fit of the model. The ROC curve and AUC statistic are quite similar to the LDA's ROC and AUC above - the ROC curve is close to the top left corner and 88.8% of the area is under the curve. It shows that the classifier is performing well with a high true-positive rate (sensitivity) and low false-negative rate (specificity).