Deng_Yehong_HW2

yd

1/31/2020

##Load packages

```r
library(ggplot2)
library(caret)
```

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

```r
library(psych)
```

```
##
## Attaching package: 'psych'

## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```r
library(rcfss)
library(tidymodels)
```

```
## -- Attaching packages -------------------------------------- tidymodels 0.0.3 --

## v broom     0.5.3     v purrr     0.3.3
## v dials     0.0.4     v recipes   0.1.9
## v dplyr     0.8.3     v rsample   0.0.5
## v infer     0.5.1     v tibble    2.1.3
## v parsnip   0.0.5     v yardstick 0.0.4

## -- Conflicts ----------------------------------------- tidymodels_conflicts() --
## x psych::%+%()           masks ggplot2::%+%()
## x scales::alpha()        masks psych::alpha(), ggplot2::alpha()
## x purrr::discard()       masks scales::discard()
## x dplyr::filter()        masks stats::filter()
## x dplyr::lag()           masks stats::lag()
## x purrr::lift()          masks caret::lift()
## x dials::margin()        masks ggplot2::margin()
## x yardstick::precision() masks caret::precision()
## x yardstick::recall()    masks caret::recall()
## x recipes::step()        masks stats::step()
## x recipes::yj_trans()    masks scales::yj_trans()
```

```r
library(pROC)
```

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

##
## Attaching package: 'pROC'

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

```r
library(e1071)
```

##The Bayes Classifier

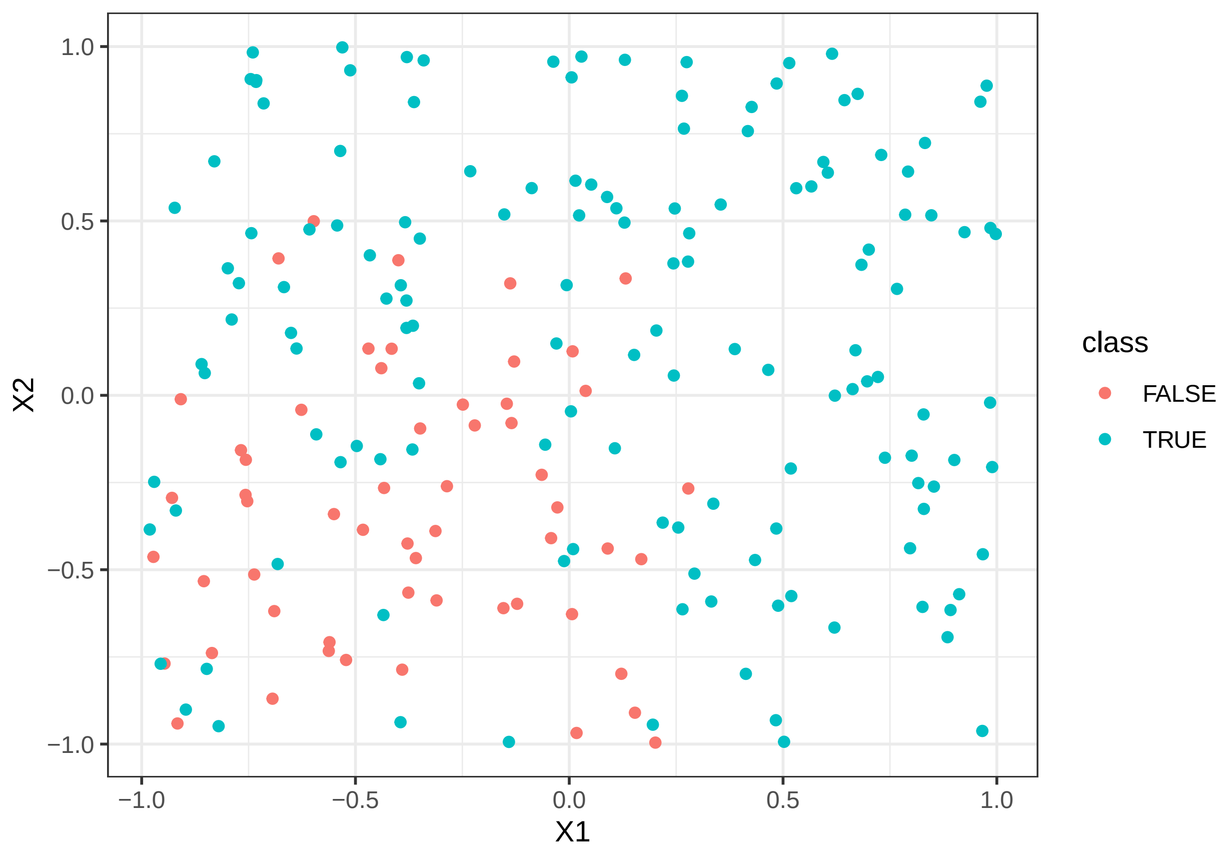
```r
#1.
#a.
set.seed(1234)
```

```r
#b.
X1 <- runif(200,-1, 1)
X2 <- runif(200,-1, 1)
df <- data.frame(X1,X2)
```

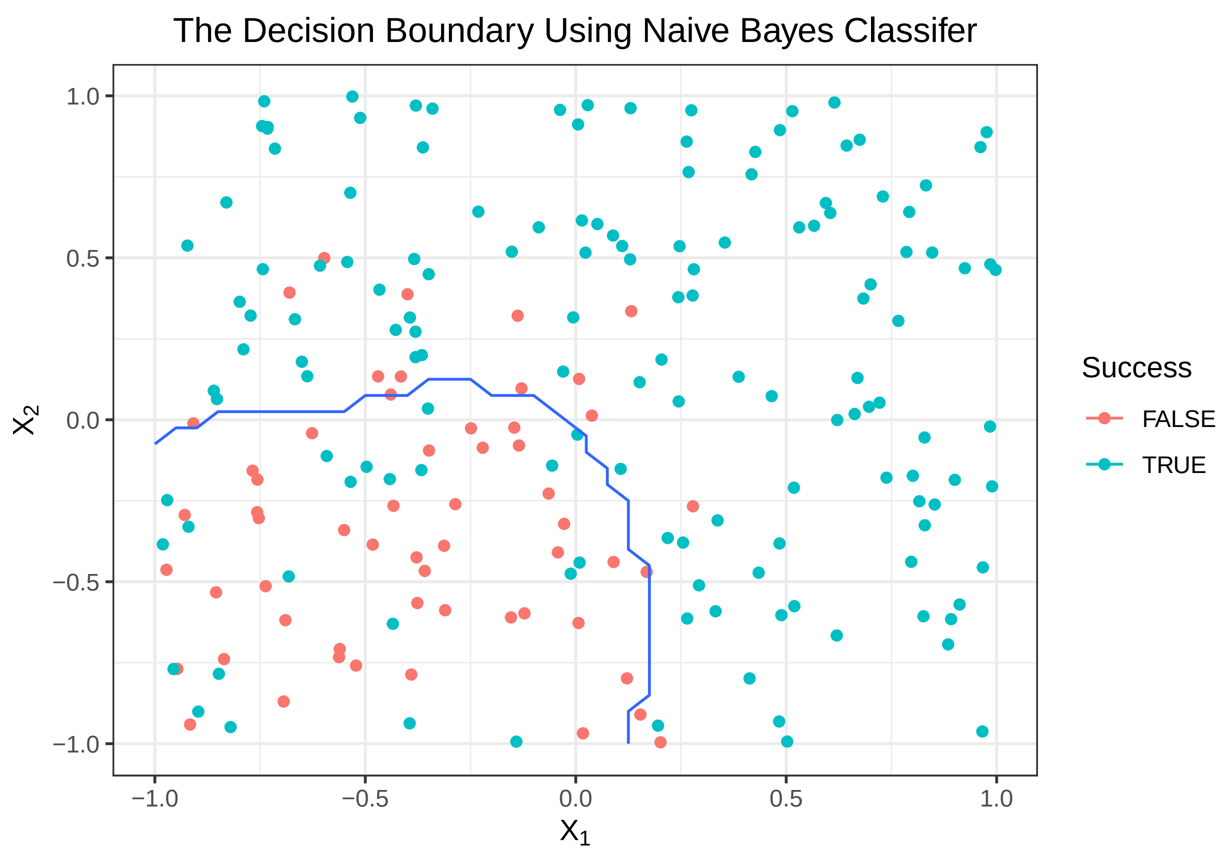
```r
#c.
err <- rnorm(200,0,0.5)
Y <- df$X1 + (df$X1)^2 + df$X2 + (df$X2)^2 + err
```

```r
#d.
odds <- exp(Y)
prob <- odds / (1 + odds)
```

```r
#e.
df$class <- as.character(prob > 0.5)
ggplot(df, aes(X1,X2, col = class))+
  geom_point() +
  theme_bw()
```



```r
#f,g.
train_control <- trainControl(
  method = "cv",
  number = 10
)
nb_model <- train(
  x = df[,c("X1","X2")],
  y = df$class,
  method = "nb",
  trControl = train_control
)
grid_range <- seq(-1, 1, by = 0.05)
grid <- expand.grid(X1 = grid_range, X2 = grid_range)
result <- predict(nb_model, newdata = grid)
grid$class <- as.numeric(result)

ggplot(df, aes(X1, X2, z=class, col=class)) +
  geom_point() +
  geom_contour(data=grid, breaks = c(1.5)) +
  theme_bw() +
  labs(color = 'Success') +
  labs(x=expression(X['1']), y=expression(X['2'])) +
  ggtitle('The Decision Boundary Using Naive Bayes Classifer') +
  theme(plot.title = element_text(hjust=.5))
```

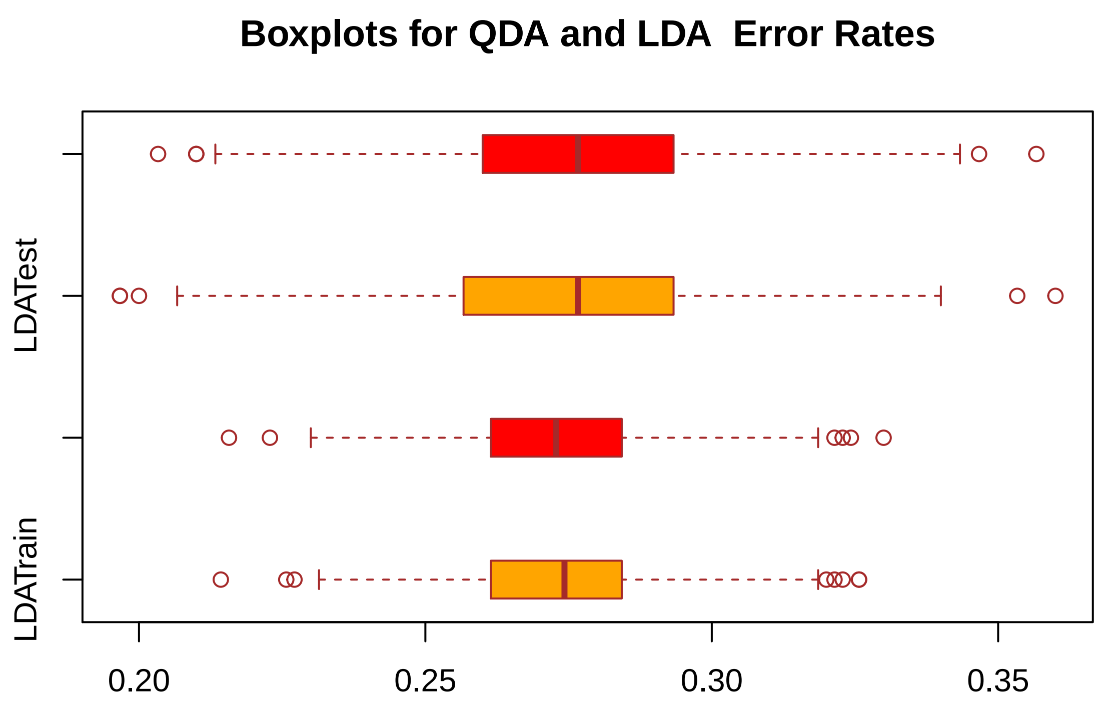


##Exploring Simulated Differences between LDA and QDA

```
#2.
##a.
lda_train_err <- vector()
lda_test_err<- vector()
qda_train_err<- vector()
qda_test_err<- vector()



for (i in 1:1000){
  x1 <- runif(1000,-1,1)
  x2 <- runif(1000,-1,1)
  y1 <- x1 + x2 + rnorm(1000, 0, 1)
  class <- y1 >= 0
  df1 <- data.frame(x1, x2,y1, class)

  split <- initial_split(df1, prop = .7)
  train <- training(split)
  test <- testing(split)

  lda_m1 <- MASS::lda(class ~ x1 + x2, data = train)
  qda_m1 <- MASS::qda(class ~ x1 + x2, data = train)

  train_lda <- predict(lda_m1, train)
  test_lda <- predict(lda_m1, test)
  train_qda <- predict(qda_m1, train)
  test_qda <- predict(qda_m1, test)

  lda_train_err <- append(lda_train_err,mean(train$class != train_lda$class))
  lda_test_err <- append(lda_test_err, mean(test$class != test_lda$class))

  qda_train_err <-append(qda_train_err, mean(train$class != train_qda$class))
  qda_test_err <- append(qda_test_err, mean(test$class != test_qda$class))
}


##b.
mydata <- data.frame(lda_train_err,qda_train_err,lda_test_err,qda_test_err)
describe(mydata)
```

```
##               vars    n mean   sd median trimmed  mad  min  max range  skew
## lda_train_err    1 1000 0.27 0.02   0.27    0.27 0.02 0.21 0.33  0.11  0.06
## qda_train_err    2 1000 0.27 0.02   0.27    0.27 0.02 0.22 0.33  0.11  0.07
## lda_test_err     3 1000 0.28 0.03   0.28    0.28 0.02 0.20 0.36  0.16 -0.04
## qda_test_err     4 1000 0.28 0.03   0.28    0.28 0.02 0.20 0.36  0.15 -0.09
##               kurtosis se
## lda_train_err     0.03  0
## qda_train_err     0.00  0
## lda_test_err     -0.29  0
## qda_test_err     -0.31  0
```

```
boxplot(lda_train_err, qda_train_err, lda_test_err, qda_test_err,
main = "Boxplots for QDA and LDA  Error Rates",
at = c(1,4,7,10),
names = c("LDATrain", "QDATrain", "LDATest", "QDATest"),
las = 0,
col = c("orange","red"),
border = "brown",
horizontal = TRUE,
notch = FALSE
)
```

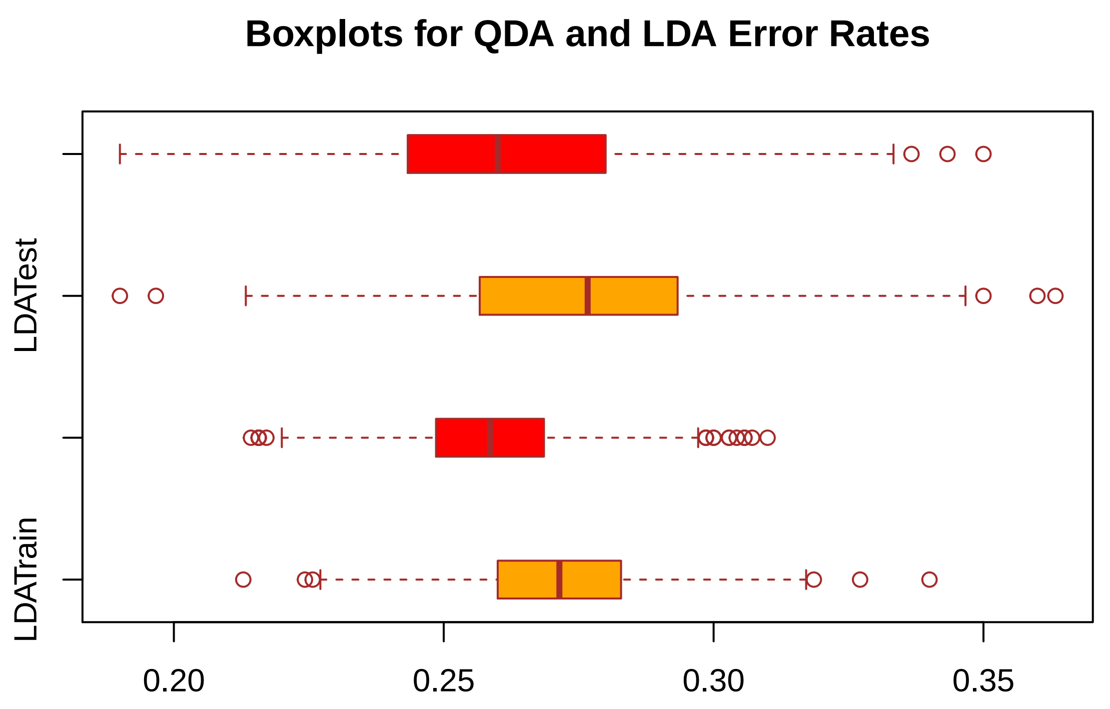


###If the Bayes decision boundary is linear, wcan see that both the QDA and LDA have similar means and median error rate on the training set and on the test set.From the descriptive table, we can see that the mean error rate on the training set for LDA is .27 and for QDA is also .27. On the test set, both QDA and LDA have the same mean of .28.Hence, it seems that QDA and LDA are equally perform when the Bayes decision boundary is linear. Nonetheless, from the boxplots, we can see that the QDA' error rate has lower variance both on the trainign set and on the test set.

```r
#3.
##a.
nl_lda_train_err <- vector()
nl_lda_test_err<- vector()
nl_qda_train_err<- vector()
nl_qda_test_err<- vector()



for (i in 1:1000){
  nlx1 <- runif(1000,-1,1)
  nlx2 <- runif(1000,-1,1)
  nly1 <- nlx1 +(nlx1)^2 + nlx2 + (nlx2)^2 + rnorm(1000, 0, 1)
  class <- nly1 >= 0
  df2 <- data.frame(nlx1, nlx2,nly1, class)

  nl_split <- initial_split(df2, prop = .7)
  nl_train <- training(nl_split)
  nl_test <- testing(nl_split)
  lda_m2 <- MASS::lda(class ~ nlx1 + nlx2, data = nl_train)
  qda_m2 <- MASS::qda(class ~ nlx1 + nlx2, data = nl_train)

  nl_train_lda <- predict(lda_m2, nl_train)
  nl_test_lda <- predict(lda_m2, nl_test)
  nl_train_qda <- predict(qda_m2, nl_train)
  nl_test_qda <- predict(qda_m2, nl_test)

  nl_lda_train_err <- append(nl_lda_train_err,mean(nl_train$class != nl_train_lda$class))
  nl_lda_test_err <- append(nl_lda_test_err, mean(nl_test$class != nl_test_lda$class))

  nl_qda_train_err <-append(nl_qda_train_err, mean(nl_train$class != nl_train_qda$class))
  nl_qda_test_err <- append(nl_qda_test_err, mean(nl_test$class != nl_test_qda$class))
}


##b.
nl_mydata <- data.frame(nl_lda_train_err,nl_qda_train_err,nl_lda_test_err,nl_qda_test_err)
describe(nl_mydata)
```

```
##                  vars    n mean   sd median trimmed  mad  min  max range skew
## nl_lda_train_err    1 1000 0.27 0.02   0.27    0.27 0.02 0.21 0.34  0.13 0.04
## nl_qda_train_err    2 1000 0.26 0.02   0.26    0.26 0.01 0.21 0.31  0.10 0.09
## nl_lda_test_err     3 1000 0.28 0.03   0.28    0.28 0.02 0.19 0.36  0.17 0.06
## nl_qda_test_err     4 1000 0.26 0.03   0.26    0.26 0.02 0.19 0.35  0.16 0.08
##                  kurtosis se
## nl_lda_train_err     0.04  0
## nl_qda_train_err    -0.05  0
## nl_lda_test_err     -0.05  0
## nl_qda_test_err     -0.17  0
```

```
boxplot(nl_lda_train_err, nl_qda_train_err, nl_lda_test_err, nl_qda_test_err,
main = "Boxplots for QDA and LDA Error Rates",
at = c(1,4,7,10),
names = c("LDATrain", "QDATrain", "LDATest", "QDATest"),
las = 0,
col = c("orange","red"),
border = "brown",
horizontal = TRUE,
notch = FALSE
)
```



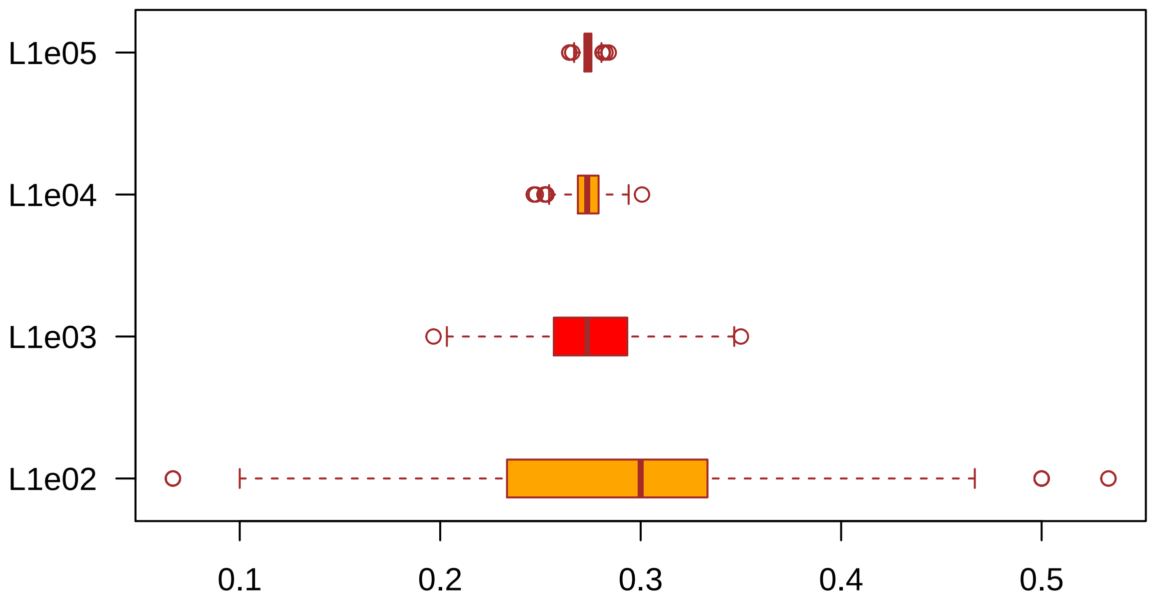
###If the Bayes decision boundary is non-linear, we can see that there is differences between the error rates for QDA and LDA both on the training set and on the test set. In general, we can perceive that the QDA perform better than the LDa. Both the descriptive statistics table and boxplot show that QDA has lower mean and median error rate on training set and test set. Therefore, when the Bayes decision boundary is non-linear, we would expect QDA to perform better both on the training set and test set.

```
#4
##a.
LDA_QDA <- function(n){
  lda_train_err <- vector()
  lda_test_err<- vector()
  qda_train_err<- vector()
  qda_test_err<- vector()
  for (i in 1:1000){
    x1 <- runif(n,-1,1)
    x2 <- runif(n,-1,1)
    y1 <- x1 + x1^2 + x2 + x2^2 + rnorm(n, 0, 1)
    class <- y1 >= 0
    df1 <- data.frame(x1, x2,y1, class)

    split <- initial_split(df1, prop = .7)
    train <- training(split)
    test <- testing(split)

    lda_m1 <- MASS::lda(class ~ x1 + x2, data = train)
    qda_m1 <- MASS::qda(class ~ x1 + x2, data = train)
    train_lda <- predict(lda_m1, train)
    test_lda <- predict(lda_m1, test)
    train_qda <- predict(qda_m1, train)
    test_qda <- predict(qda_m1, test)

    lda_train_err <- append(lda_train_err,mean(train$class != train_lda$class))
    lda_test_err <- append(lda_test_err, mean(test$class != test_lda$class))

    qda_train_err <-append(qda_train_err, mean(train$class != train_qda$class))
    qda_test_err <- append(qda_test_err, mean(test$class != test_qda$class))
  }
  fin_df <- data.frame(lda_train_err, qda_train_err, lda_test_err, qda_test_err)
  return(fin_df)
}

n_100 <- LDA_QDA(100)

n_1000<- LDA_QDA(1000)

n_10000<- LDA_QDA(10000)

n_100000<- LDA_QDA(100000)

##b.
lda_data <- data.frame(n_100$lda_test_err, n_1000$lda_test_err, n_10000$lda_test_err, n_100000$lda_test_
describe(lda_data)
```

```
##                        vars    n mean   sd median trimmed  mad  min  max range
## n_100.lda_test_err        1 1000 0.29 0.08   0.30    0.29 0.10 0.07 0.53  0.47
## n_1000.lda_test_err       2 1000 0.27 0.03   0.27    0.27 0.02 0.20 0.35  0.15
## n_10000.lda_test_err      3 1000 0.27 0.01   0.27    0.27 0.01 0.25 0.30  0.05
## n_100000.lda_test_err     4 1000 0.27 0.00   0.27    0.27 0.00 0.26 0.28  0.02
##                         skew kurtosis se
## n_100.lda_test_err      0.14    -0.14  0
## n_1000.lda_test_err     0.09    -0.06  0
## n_10000.lda_test_err   -0.06    -0.04  0
## n_100000.lda_test_err   0.01     0.15  0
```

```
boxplot(n_100$lda_test_err, n_1000$lda_test_err, n_10000$lda_test_err, n_100000$lda_test_err,
main = "Fig 4.1 Boxplots for LDA Errors accross Sample Sizes",
at = c(1,4,7,10),
names = c("L1e02", "L1e03", "L1e04", "L1e05"),
las = 1,
col = c("orange","red"),
border = "brown",
horizontal = TRUE,
notch = FALSE
)
```

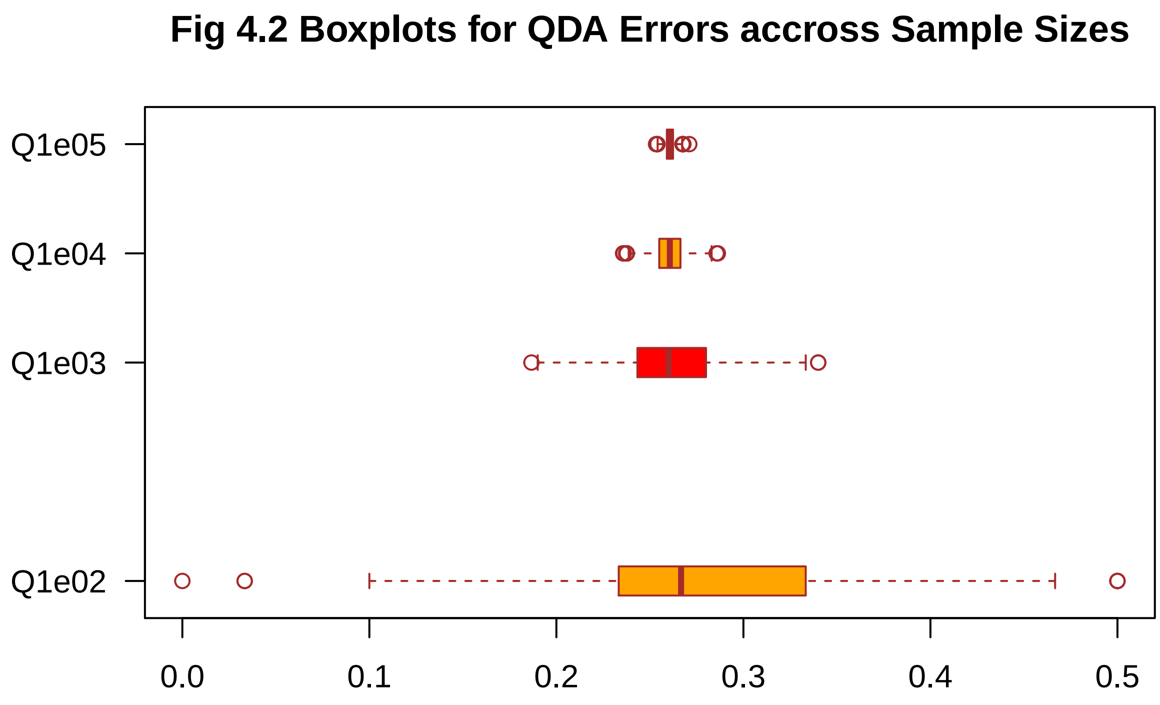
Fig 4.1 Boxplots for LDA Errors accross Sample Sizes



```
qda_data <- data.frame(n_100$qda_test_err, n_1000$qda_test_err, n_10000$qda_test_err, n_100000$qda_test_
describe(qda_data)
```

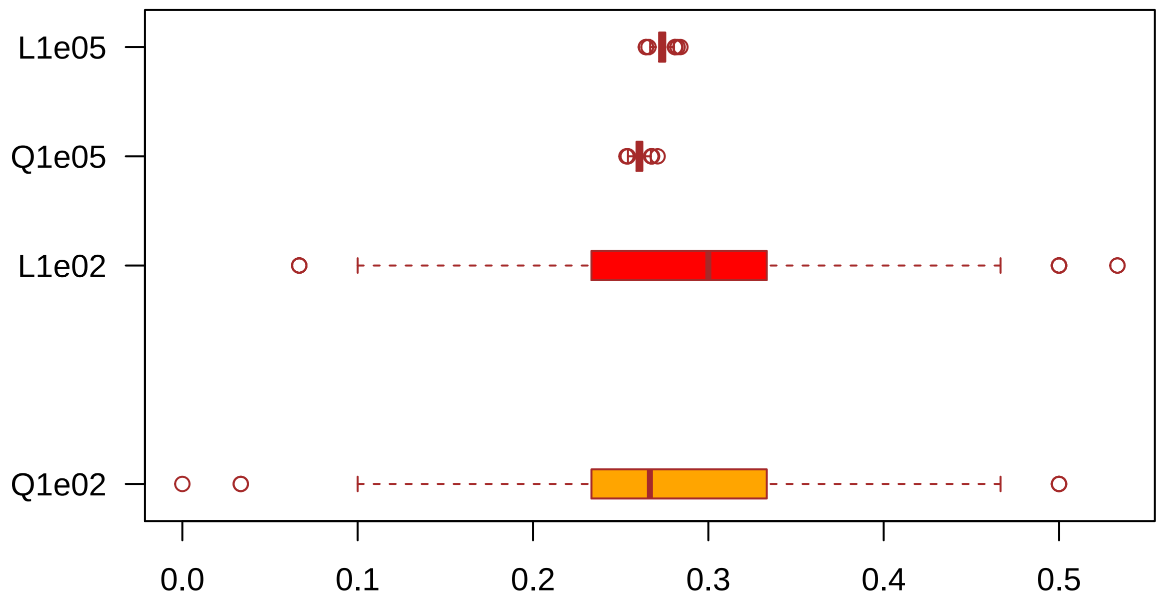
```
##                         vars    n mean   sd median trimmed  mad  min  max range
## n_100.qda_test_err         1 1000 0.28 0.08   0.27    0.28 0.10 0.00 0.50  0.50
## n_1000.qda_test_err        2 1000 0.26 0.03   0.26    0.26 0.02 0.19 0.34  0.15
## n_10000.qda_test_err       3 1000 0.26 0.01   0.26    0.26 0.01 0.24 0.29  0.05
## n_100000.qda_test_err      4 1000 0.26 0.00   0.26    0.26 0.00 0.25 0.27  0.02
##                          skew kurtosis se
## n_100.qda_test_err       0.15    -0.12  0
## n_1000.qda_test_err      0.10    -0.22  0
## n_10000.qda_test_err    -0.06    -0.04  0
## n_100000.qda_test_err    0.06    -0.03  0
```

```
boxplot(n_100$qda_test_err, n_1000$qda_test_err, n_10000$qda_test_err, n_100000$qda_test_err,
main = "Fig 4.2 Boxplots for QDA Errors accross Sample Sizes",
at = c(1,7,10,13),
names = c("Q1e02", "Q1e03", "Q1e04", "Q1e05"),
las = 1,
col = c("orange","red"),
border = "brown",
horizontal = TRUE,
notch = FALSE
)
```



```
boxplot(n_100$qda_test_err, n_100$lda_test_err, n_100000$qda_test_err, n_100000$lda_test_err,
main = "Fig 4.3 Boxplots for Comparision for QDA and LDA",
at = c(1,7,10,13),
names = c("Q1e02", "L1e02", "Q1e05", "L1e05"),
las = 1,
col = c("orange","red"),
border = "brown",
horizontal = TRUE,
notch = FALSE
)
```

Fig 4.3 Boxplots for Comparision for QDA and LDA



###As we can see in Fig 4.1 and 4.2, the median test error rate for both QDA and LDA decreased as the sample became bigger. Moreover, the variances for test error also declined. From Fig 4.3 we can see that as the sample become bigger, the median of test error rate for QDA and LDA become more and more similar. Therefore, when the sample size becomes bigger, we can expect that the test error rate of QDA relative to LDA to improve.

##Modeling voter turnout

```
#5
##a.

mental_health <- read.csv("C:/Users/brobo/Desktop/r stuff/problem-set-2-master/mental_health.csv", head
mental_health<- na.omit(mental_health)

mh_split<- initial_split(mental_health, prop = 0.7)
mh_train <- training(mh_split)
mh_test <- testing(mh_split)
```

```
##b.
#I.
mh_logit <- glm(vote96 ~ ., data = mh_train, family = binomial)
#II.
mh_lda <- MASS::lda(vote96 ~ ., data = mh_train)
#III.
mh_qda <- MASS::qda(vote96 ~ ., data = mh_train)
#IV.
x_nb <- mh_train%>%
```

```r
  select(-vote96)
y_nb <- mh_train$vote96z
mh_nb <- naiveBayes(vote96~., data = mh_train)
#V.
mh_knn <- tibble(k = 1:10,
                 knn_train = map(k, ~ class::knn(select(mh_train, -vote96),
                                                test = select(mh_test, -vote96),
                                                cl = mh_train$vote96, k = .)),
                 knn_test = map(k, ~ attr(class::knn(select(mh_train, -vote96),
                                               test = select(mh_test, -vote96),
                                               cl = mh_train$vote96, k = .,
                                               prob = TRUE),'prob')),
                err_test = map_dbl(knn_test, ~ mean(mh_test$vote96 != .)))
```

```r
##c.
pred_tests <- tibble(log = predict(mh_logit, mh_test),
                     lda = predict(mh_lda, mh_test)$posterior[,1],
                     qda = predict(mh_qda, mh_test)$posterior[,1],
                     bayes = predict(mh_nb, mh_test, type = 'raw')[,1])

#log error
log_class <- pred_tests$log >= 0.5
log_err <- mean(mh_test$vote96 != log_class)
#lda error
lda_class <- pred_tests$log >= 0.5
lda_err <- mean(mh_test$vote96 != lda_class)
#qda error
qda_class <- pred_tests$qda >= 0.5
qda_err <- mean(mh_test$vote96 != qda_class)
#naive bayes error
nb_class <- pred_tests$bayes >= 0.5
nb_err <- mean(mh_test$vote96 != nb_class)
#knn error
knn_err <- tibble(mh_knn$err_test)

err_df <- data.frame(log_err,lda_err,qda_err,nb_err)
```

```r
knitr::kable(err_df)
```

log_err	lda_err	qda_err	nb_err
0.269341	0.269341	0.6762178	0.7020057

```r
knitr::kable(knn_err)
```

mh_knn$err_test
0.3438395
0.5587393
0.6762178
0.7163324
0.7421203

mh_knn$err_test
0.7851003
0.8051576
0.8309456
0.8481375
0.8796562

```r
#log,lda,qda,nb roc
par(pty = "s")
log_roc <- roc(mh_test$vote96,pred_tests$log,plot = TRUE, legacy.axes = TRUE, col = "blue")
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls < cases
```

```r
lda_roc <- plot.roc(mh_test$vote96, pred_tests$lda, col = "green",add = TRUE)
```

```
## Setting levels: control = 0, case = 1

## Setting direction: controls > cases
```

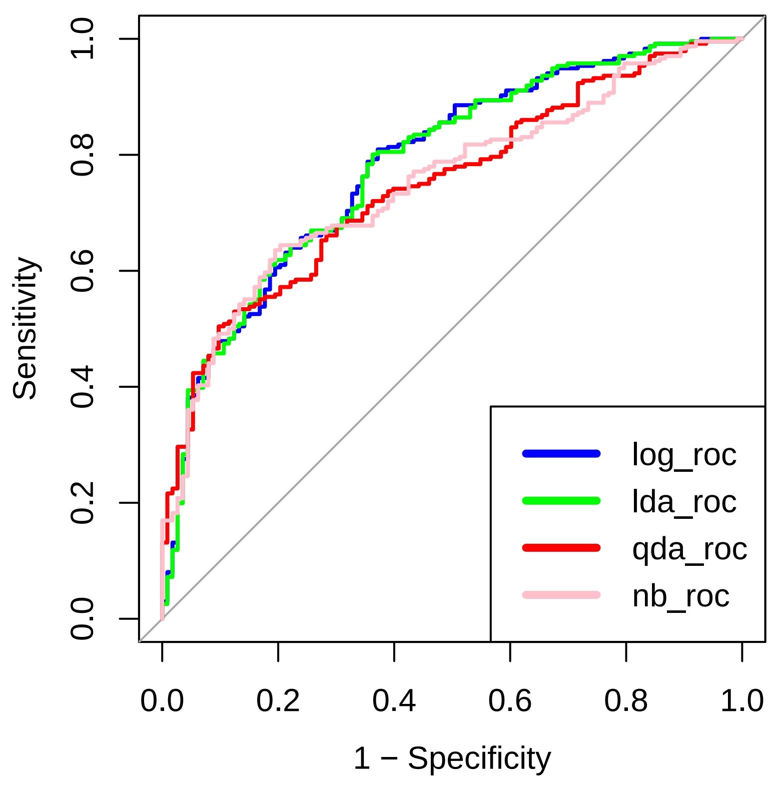
```r
qda_roc <- plot.roc(mh_test$vote96, pred_tests$qda, col = "red", add = TRUE)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls > cases
```

```r
nb_roc <- plot.roc(mh_test$vote96, pred_tests$bayes, col = "pink", add = TRUE)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls > cases
```

```r
legend("bottomright", legend = c("log_roc","lda_roc","qda_roc","nb_roc"), col = c("blue","green", "red"
```



```
#knn roc
knn_test <- as_tibble(mh_knn$knn_test, .name_repair = 'unique')
```

```
## New names:
## * `` -> ...1
## * `` -> ...2
## * `` -> ...3
## * `` -> ...4
## * `` -> ...5
## * ... and 5 more problems
```

```
par(pty = "s")
knn1_roc <-roc(mh_test$vote96, as.numeric(knn_test$...1),plot = TRUE, legacy.axes = TRUE, col = "#DC143
```

```
## Setting levels: control = 0, case = 1
```

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

```
knn2_roc <- plot.roc(mh_test$vote96, as.numeric(knn_test$...2), col = "#87CEEB", add = TRUE)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
knn3_roc <- plot.roc(mh_test$vote96, as.numeric(knn_test$...3), col = "#00FF00", add = TRUE)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
knn4_roc <- plot.roc(mh_test$vote96, as.numeric(knn_test$...4), col = "#F0E68C", add = TRUE)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
knn5_roc <- plot.roc(mh_test$vote96, as.numeric(knn_test$...5), col = "#4B0082", add = TRUE)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
knn6_roc <- plot.roc(mh_test$vote96, as.numeric(knn_test$...6), col = "#4169E1", add = TRUE)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
knn7_roc <- plot.roc(mh_test$vote96, as.numeric(knn_test$...7), col = "#00FFFF", add = TRUE)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
knn8_roc <-plot.roc(mh_test$vote96, as.numeric(knn_test$...8), col = "#ADFF2F", add = TRUE)
```

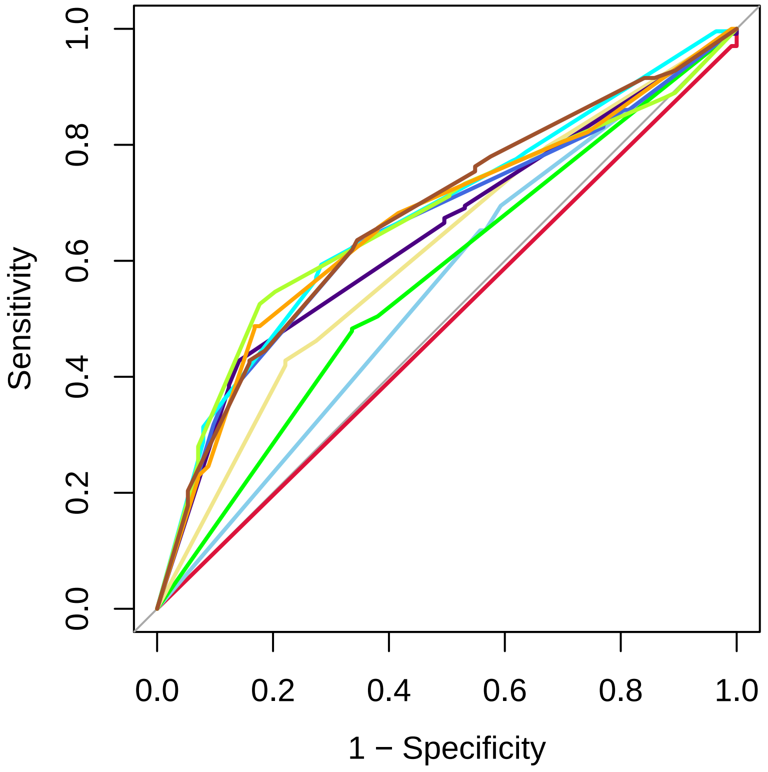
```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
knn9_roc <- plot.roc(mh_test$vote96, as.numeric(knn_test$...9), col = "#FFA500", add = TRUE)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
knn10_roc <- plot.roc(mh_test$vote96, as.numeric(knn_test$...10), col = "#A0522D", add = TRUE)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```



```r
#auc
auc_df <- tibble(log_roc$auc, lda_roc$auc, qda_roc$auc, nb_roc$auc, knn1_roc$auc, knn2_roc$auc, knn3_ro
auc_df
```

```
## # A tibble: 1 x 14
##   `log_roc$auc` `lda_roc$auc` `qda_roc$auc` `nb_roc$auc` `knn1_roc$auc`
##           <dbl>         <dbl>         <dbl>        <dbl>          <dbl>
## 1         0.783         0.783         0.748        0.750          0.489
## # ... with 9 more variables: `knn2_roc$auc` <dbl>, `knn3_roc$auc` <dbl>,
## #   `knn4_roc$auc` <dbl>, `knn5_roc$auc` <dbl>, `knn6_roc$auc` <dbl>,
## #   `knn7_roc$auc` <dbl>, `knn8_roc$auc` <dbl>, `knn9_roc$auc` <dbl>,
## #   `knn10_roc$auc` <dbl>
```

##d ###To choose the "best" model, we can first check the AUC for each model. Among all the models, the AUC for logistic regression model and LDA model have the highest AUC. To further evaluate the model, we can see that the logistic regression model and LDA model also have the lowest error rate. Therefore, since the AUC for LDA model is slightly higher than AUC for the logistic regression model, we can conclude that LDA model is the best model.