HW2 Responses

Yawei LI

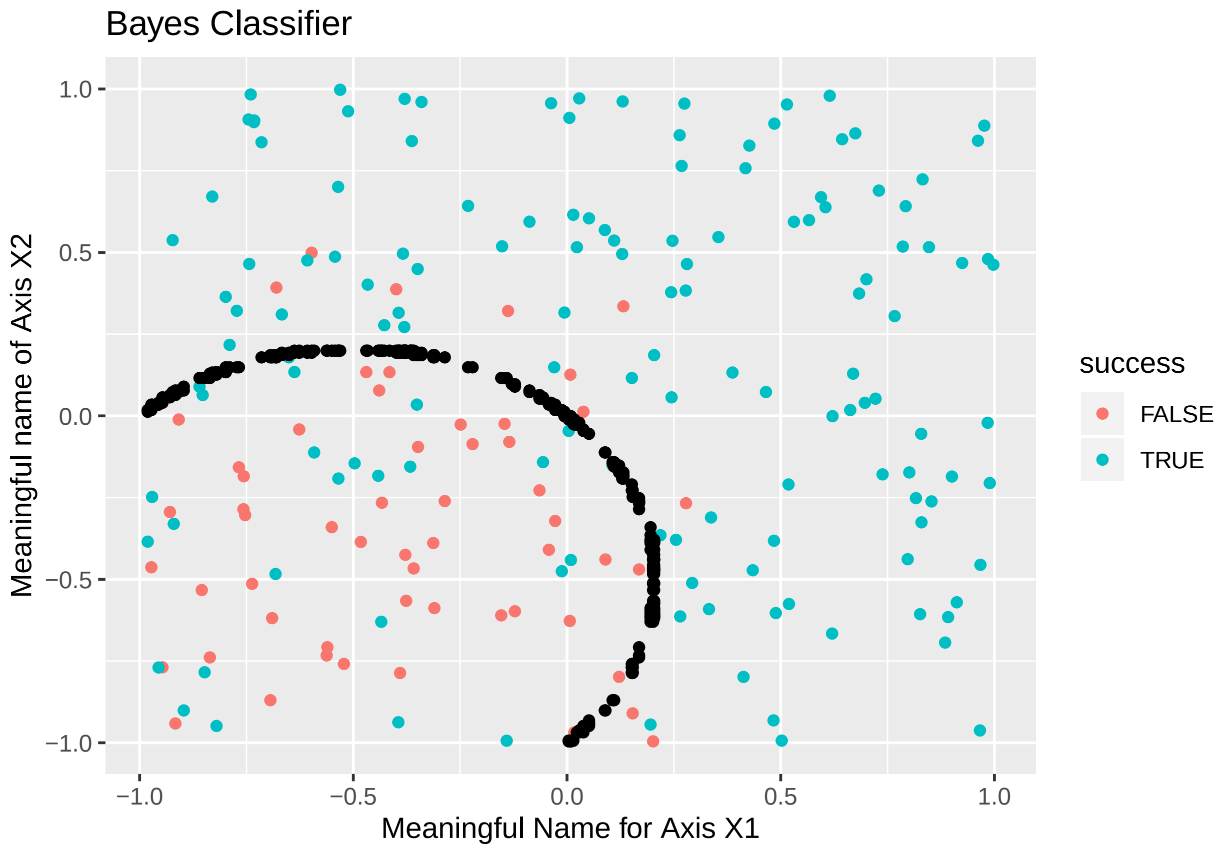
2020/2/2

The Bayes Classfier

Question 1

Graph is produced below. Black dots mark the Bayes decision boundary.

```
set.seed(1234)
x1 <- runif(200, min = -1, max = 1)
x2 <- runif(200, min = -1, max = 1)
epsilon <- rnorm(200,0,0.5)
y <- x1^2 + x2^2 + x1 + x2 + epsilon
odds <- exp(y)
pr <- odds/(1+odds)
success <- as.factor(pr > 0.5)
df = data.frame(x1,x2,success)
scatter = ggplot() +
  geom_point(aes(x=x1,y=x2,color = success),data = df) +
  labs(title = 'Bayes Classifier',x = 'Meaningful Name for Axis X1', y =
               'Meaningful name of Axis X2')
x1_b = numeric(0)
x2_b = numeric(0)
for (a in x1){
  for(b in x2){
    if (abs(a^2 + a + b^2 + b) <= 0.01){

        x1_b <- append(x1_b,a)
        x2_b <- append(x2_b,b)
      }
  }
}

scatter + geom_point(aes(x = x1_b,y = x2_b))
```



LDA & QDA

Question 2

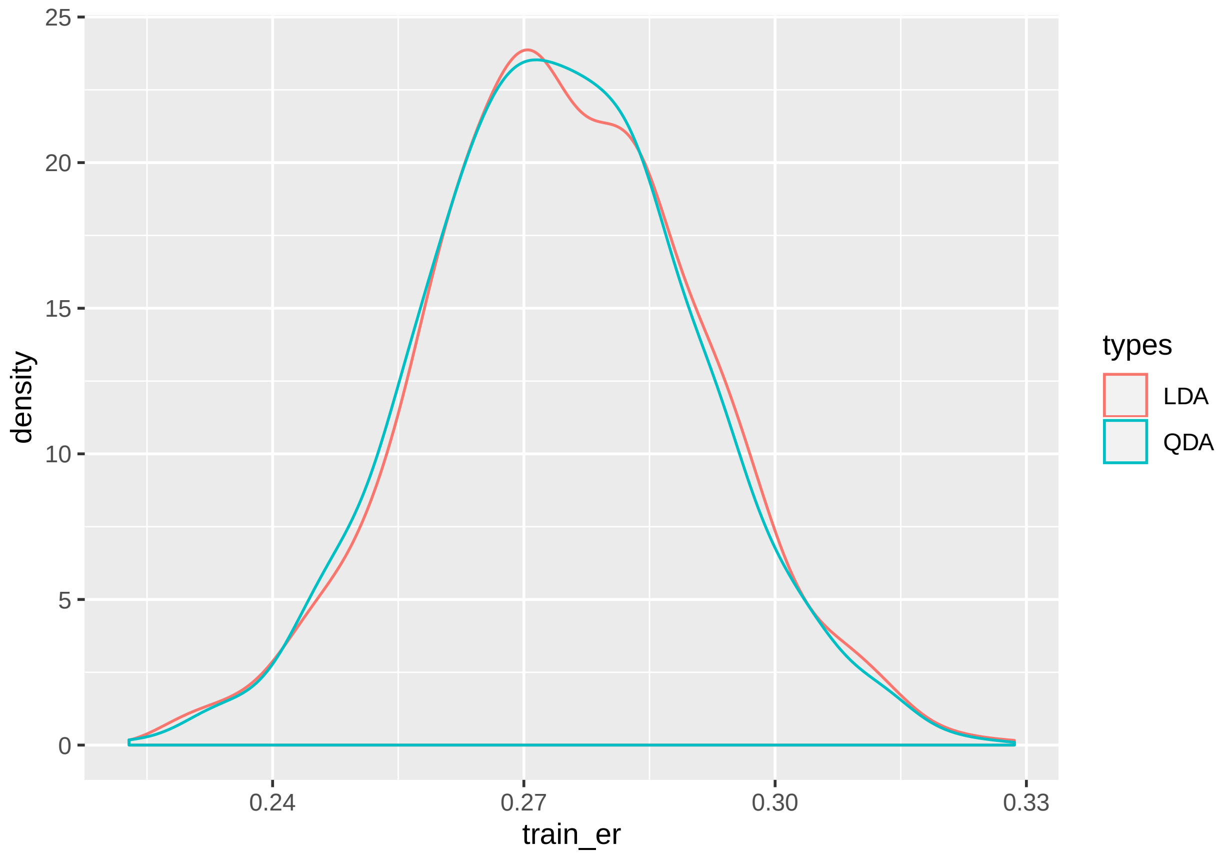
On the train set, QDA yields a smaller error rate because its flexibility makes it possible for it to fit train data better. On the test set, LDA yields a slightly smaller error rate because QDA may overfit the testing set. Still, graphs and tables show that the differences are rather small.

```
set.seed(3234)

lda_error_train = numeric(0)
qda_error_train = numeric(0)
lda_error_test = numeric(0)
qda_error_test = numeric(0)

for (i in 1:1000){
  x1 <- runif(1000, min = -1, max = 1)
  x2 <- runif(1000, min = -1, max = 1)
  epsilon <- rnorm(1000,0,1)
  y <- as.factor((x1 + x2 + epsilon) > 0)
  df <- data.frame(x1,x2,y,epsilon)
  split <- initial_split(df, prop = .7)
  train <- training(split)
  test <- testing(split)
  lda_m1 <- MASS::lda(y ~ x1 + x2, data = train)
```

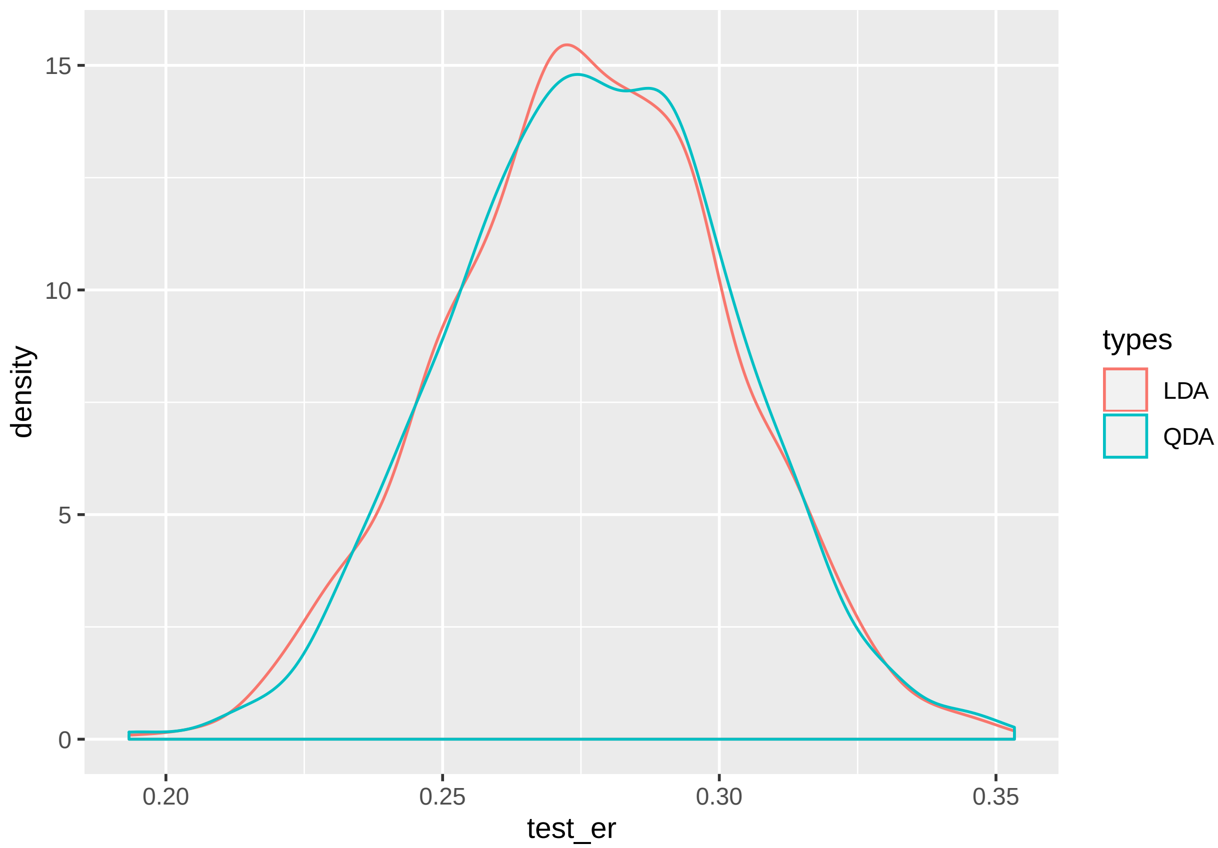
```r
  qda_m1 <- MASS::qda(y ~ x1 + x2, data = train)
  pred_lda_train = predict(lda_m1, newdata = train)
  pred_lda_test = predict(lda_m1, newdata = test)
  pred_qda_train = predict(qda_m1, newdata = train)
  pred_qda_test = predict(qda_m1, newdata = test)
  cm_lda_train <- confusionMatrix(pred_lda_train$class, train$y)
  cm_lda_test <- confusionMatrix(pred_lda_test$class, test$y)
  cm_qda_train <- confusionMatrix(pred_qda_train$class, train$y)
  cm_qda_test <- confusionMatrix(pred_qda_test$class, test$y)
  lda_error_train[i] = 1 - cm_lda_train$overall[1]
  lda_error_test[i] = 1 - cm_lda_test$overall[1]
  qda_error_train[i] = 1 - cm_qda_train$overall[1]
  qda_error_test[i] = 1 - cm_qda_test$overall[1]
}
train_error_rate = c(mean(lda_error_train),mean(qda_error_train))
test_error_rate = c(mean(lda_error_test),mean(qda_error_test))
error_table = data.frame(train_error_rate,test_error_rate,row.names=
                           c('LDA','QDA'))
print(error_table)
```

```
##     train_error_rate test_error_rate
## LDA        0.2746414       0.2763667
## QDA        0.2740700       0.2770433
```

```r
types <- c(rep('LDA',1000),rep('QDA',1000))
train_er <- c(lda_error_train,qda_error_train)
train_table <- data.frame(types,train_er)
p1 <- ggplot(train_table,aes(x = train_er,color = types)) + geom_density()

test_er <- c(lda_error_test,qda_error_test)
test_table <- data.frame(types,test_er)
p2 <- ggplot(test_table,aes(x = test_er,color = types)) + geom_density()
p1
```




```
p2
```



Question 3

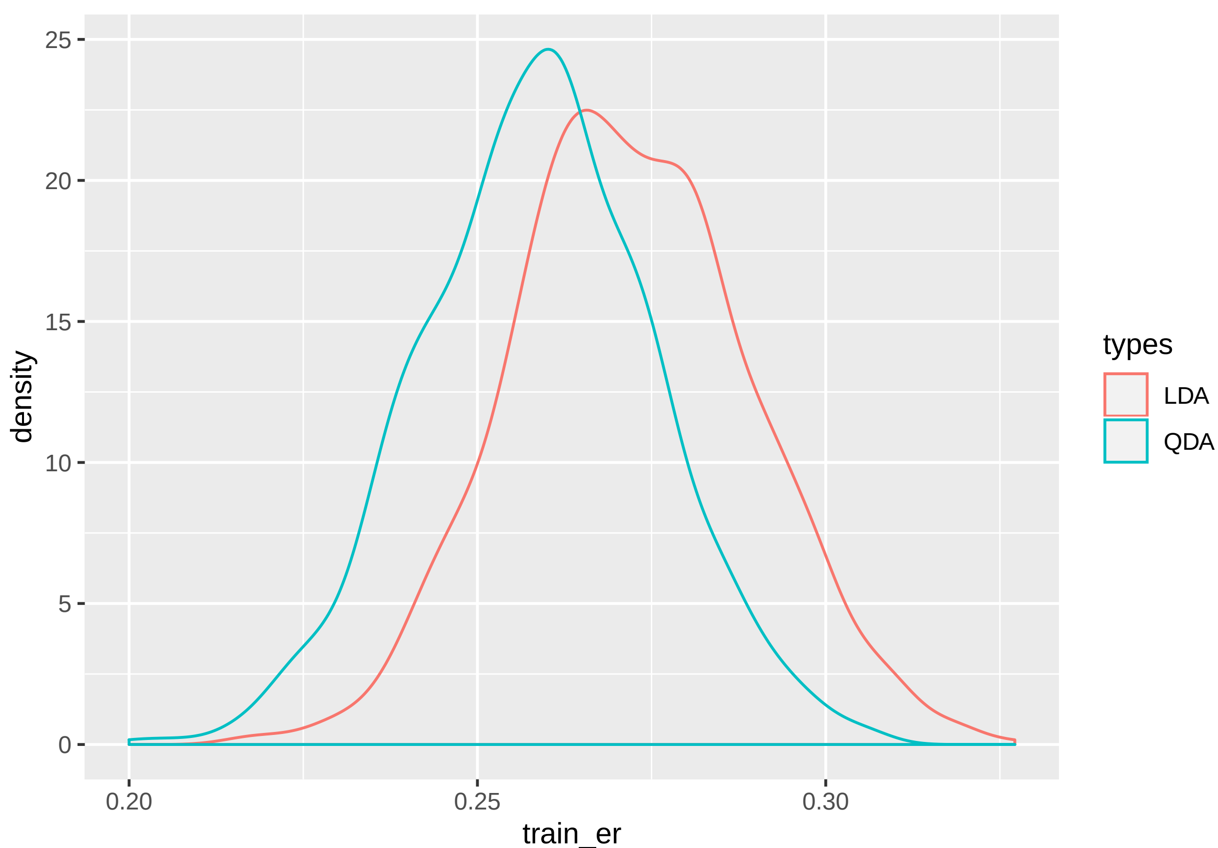
As can be seen from the tables and graphs, QDA model performs better in both training and test set, yielding a smaller error rate due to its flexibility capturing non-linear relationships.

```
set.seed(3234)

lda_error_train = numeric(0)
qda_error_train = numeric(0)
lda_error_test = numeric(0)
qda_error_test = numeric(0)

for (i in 1:1000){
  x1 <- runif(1000, min = -1, max = 1)
  x2 <- runif(1000, min = -1, max = 1)
  epsilon <- rnorm(1000,0,1)
  y <- as.factor((x1^2 + x1 + x2^2 + x2 + epsilon) > 0)
  df <- data.frame(x1,x2,y,epsilon)
  split <- initial_split(df, prop = .7)
  train <- training(split)
  test <- testing(split)
  lda_m1 <- MASS::lda(y ~ x1 + x2, data = train)
  qda_m1 <- MASS::qda(y ~ x1 + x2, data = train)
  pred_lda_train = predict(lda_m1, newdata = train)
  pred_lda_test = predict(lda_m1, newdata = test)
```

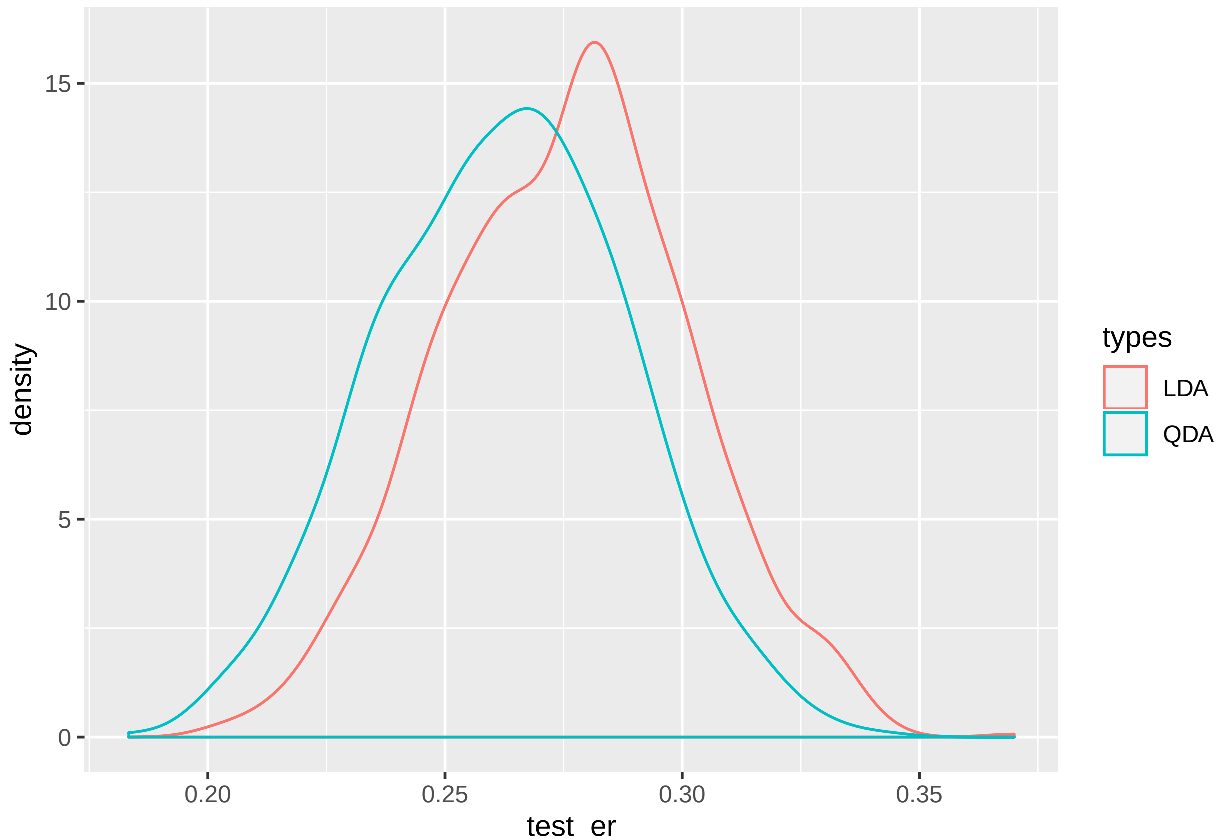
```
  pred_qda_train = predict(qda_m1, newdata = train)
  pred_qda_test = predict(qda_m1, newdata = test)
  cm_lda_train <- confusionMatrix(pred_lda_train$class, train$y)
  cm_lda_test <- confusionMatrix(pred_lda_test$class, test$y)
  cm_qda_train <- confusionMatrix(pred_qda_train$class, train$y)
  cm_qda_test <- confusionMatrix(pred_qda_test$class, test$y)
  lda_error_train[i] = 1 - cm_lda_train$overall[1]
  lda_error_test[i] = 1 - cm_lda_test$overall[1]
  qda_error_train[i] = 1 - cm_qda_train$overall[1]
  qda_error_test[i] = 1 - cm_qda_test$overall[1]
}
train_error_rate = c(mean(lda_error_train),mean(qda_error_train))
test_error_rate = c(mean(lda_error_test),mean(qda_error_test))
error_table = data.frame(train_error_rate,test_error_rate,row.names=
                           c('LDA','QDA'))
print(error_table)
```

```
##     train_error_rate test_error_rate
## LDA        0.2718714       0.2755533
## QDA        0.2586014       0.2622467
```

```
types <- c(rep('LDA',1000),rep('QDA',1000))
train_er <- c(lda_error_train,qda_error_train)
train_table <- data.frame(types,train_er)
p1 <- ggplot(train_table,aes(x = train_er,color = types)) + geom_density()
p1
```



```r
test_er <- c(lda_error_test,qda_error_test)
test_table <- data.frame(types,test_er)
p2 <- ggplot(test_table,aes(x = test_er,color = types)) + geom_density()
p2
```



Question 4

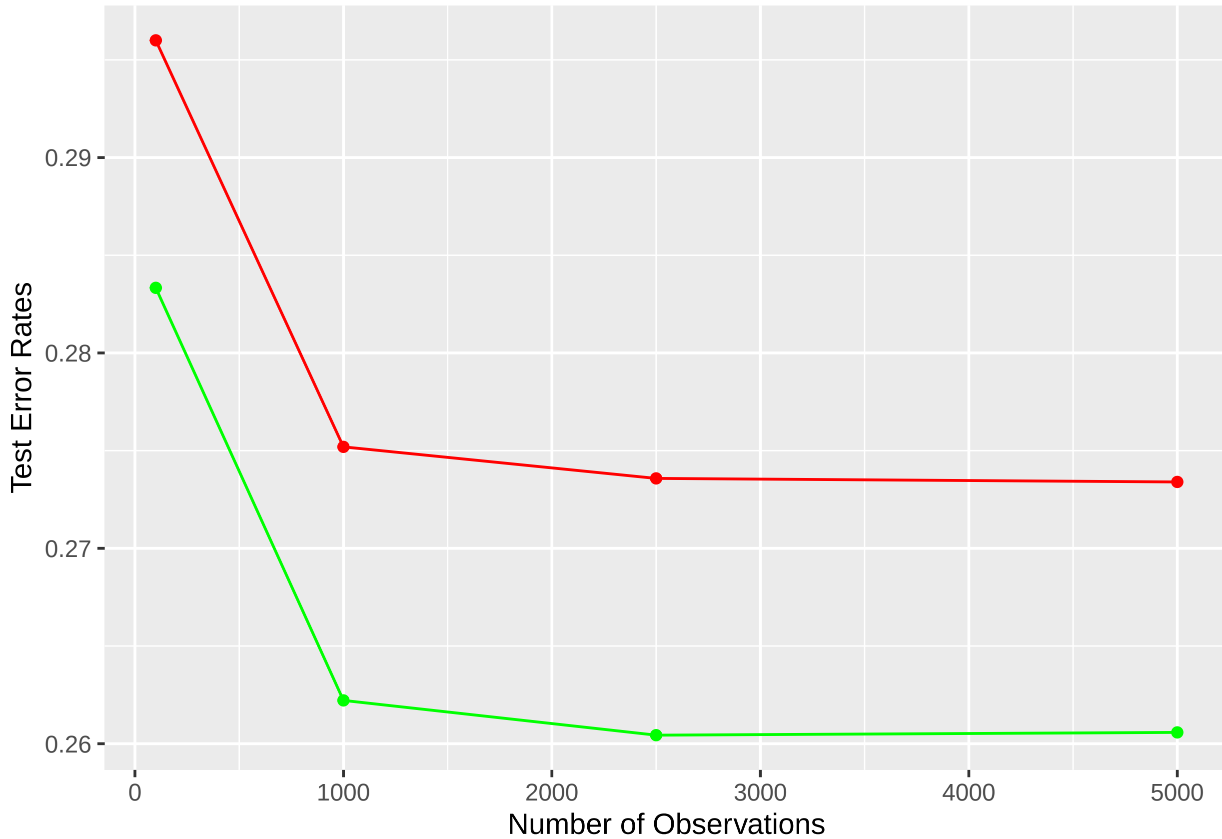
As can be seen from the plot, the relative performance of QDA models (Red) become better than LDA models (Green) as the number of observations increase, because when n is large, the impact from variance decreases, making QDA models work better.

```r
set.seed(4234)
get_results <- function(n,sim = 1000)
#Input:
#  n(int):Number of Observations.
#  sim(int):Number of Simulations.
#Returns: a List of error rate tables, training error rate density plot and test
#  error rate density plot.
{
lda_error_train = numeric(0)
qda_error_train = numeric(0)
lda_error_test = numeric(0)
qda_error_test = numeric(0)

for (i in 1:n){
```

```r
  x1 <- runif(n, min = -1, max = 1)
  x2 <- runif(n, min = -1, max = 1)
  epsilon <- rnorm(n,0,1)
  y <- as.factor((x1^2 + x1 + x2^2 + x2 + epsilon) > 0)
  df <- data.frame(x1,x2,y,epsilon)
  split <- initial_split(df, prop = .7)
  train <- training(split)
  test <- testing(split)
  lda_m1 <- MASS::lda(y ~ x1 + x2, data = train)
  qda_m1 <- MASS::qda(y ~ x1 + x2, data = train)
  pred_lda_test = predict(lda_m1, newdata = test)
  pred_qda_test = predict(qda_m1, newdata = test)
  cm_lda_test <- confusionMatrix(pred_lda_test$class, test$y)
  cm_qda_test <- confusionMatrix(pred_qda_test$class, test$y)
  lda_error_test[i] = 1 - cm_lda_test$overall[1]
  qda_error_test[i] = 1 - cm_qda_test$overall[1]
}

return(c(mean(lda_error_test),mean(qda_error_test)))

}

n100  <- get_results(100)
n1000 <- get_results(1000)
n2500 <- get_results(2500)
n5000 <- get_results(5000)

lda_er <- c(n100[1],n1000[1],n2500[1],n5000[1])
qda_er <- c(n100[2],n1000[2],n2500[2],n5000[2])

df <- data.frame(lda_er,qda_er,row.names = c('N=100','N=1000','N=2500','N=5000'))

print(df)
```

```
##           lda_er    qda_er
## N=100  0.2960000 0.2833333
## N=1000 0.2751933 0.2622167
## N=2500 0.2735792 0.2604373
## N=5000 0.2733972 0.2605783
```

```r
lines <- ggplot() +
  geom_line(aes(y = lda_er,x = c(100,1000,2500,5000)),colour = 'Red') +
  geom_point(aes(y = lda_er,x = c(100,1000,2500,5000)),colour = 'Red') +
  geom_line(aes(y = qda_er,x = c(100,1000,2500,5000)),colour = 'Green') +
  geom_point(aes(y = qda_er,x = c(100,1000,2500,5000)),colour = 'Green') +
  labs(x = 'Number of Observations',y = 'Test Error Rates')

lines
```



Question 5

As can be seen from the table, the Logit model performs the best, having highest AUC and lowest error rate.

```
# Data
set.seed(6234)
data <- read.csv('mental_health.csv')
data <- na.omit(data)
mhsplit <- initial_split(data, prop = .7)
mhtrain <- training(mhsplit)
mhtest <- testing(mhsplit)
turnout <- as.factor(as.logical(mhtest$vote96))

# Models
logit_mh <- glm(vote96 ~ mhealth_sum + age + educ + black + female
               + married + inc10 , data = mhtrain, family=binomial(link="logit"))
logit_test <- predict(logit_mh,newdata = mhtest)
logit_test_factor <- as.factor(logit_test > 0.5)
logit_cm <- confusionMatrix(logit_test_factor,turnout)
logit_er <- 1 - logit_cm$overall[1]
lda_mh <- MASS::lda(vote96 ~ mhealth_sum + age + educ + black + female
               + married + inc10, data = mhtrain)
lda_test <- predict(lda_mh,newdata = mhtest)
lda_cm <- confusionMatrix(unlist(lda_test[1]),as.factor(mhtest$vote96))
lda_er <- 1 - lda_cm$overall[1]
```

```r
qda_mh <- MASS::qda(vote96 ~ mhealth_sum + age + educ + black + female
               + married + inc10, data = mhtrain)
qda_test <- predict(qda_mh,newdata = mhtest)
qda_cm <- confusionMatrix(unlist(qda_test[1]),as.factor(mhtest$vote96))
qda_er <- 1 - qda_cm$overall[1]
naive_mh <- naiveBayes(vote96 ~ mhealth_sum + age + educ + black + female
               + married + inc10, data = mhtrain)
naive_test <- predict(naive_mh,newdata = mhtest,type = 'raw')
naive_cm <- confusionMatrix(as.factor(naive_test[,2] > 0.5),turnout)
naive_er <- 1 - naive_cm$overall[1]

knn1_mh <- knn3Train(mhtrain,mhtest,mhtrain$vote96,k=1)

mse_knn <- tibble(k = 1:10,
                  knn_train = map(k, ~ class::knn(dplyr::select(mhtrain, -vote96),
                                                  test = dplyr::select(mhtrain, -vote96),
                                                  cl = mhtrain$vote96, k = .)),
                  knn_test = map(k, ~ class::knn(dplyr::select(mhtrain, -vote96),
                                                 test = dplyr::select(mhtest, -vote96),
                                                 cl = mhtrain$vote96, k = .)),
                  err_train = map_dbl(knn_train, ~ mean(mhtest$vote96 != .)),
                  err_test = map_dbl(knn_test, ~ mean(mhtest$vote96 != .)))

# Metrics
logit_auc <- roc(mhtest$vote96,logit_test)$auc[1]
lda_auc <- roc(mhtest$vote96,as.numeric(unlist(lda_test[1])) - 1)$auc[1]
qda_auc <- roc(mhtest$vote96,as.numeric(unlist(qda_test[1])) - 1)$auc[1]
naive_auc <- roc(mhtest$vote96,as.numeric(naive_test[,2]>0.5))$auc[1]
knn_auc <- numeric(0)
knn_er <- mse_knn$err_test
for (i in 1:10){
  knn_auc[i] <- roc(mhtest$vote96,as.numeric(mse_knn$knn_test[[i]])-1)[[9]][1]
}

auc <- append(c(logit_auc,lda_auc,qda_auc,naive_auc),knn_auc)
er <- append(c(logit_er,lda_er,qda_er,naive_er),knn_er)
df <- data.frame(auc,er,row.names = c('Logit','LDA','QDA','Naive Bayes',
      'KNN w/ k = 1','k=2','k=3','k=4','k=5','k=6','k=7','k=8','k=9','k=10'))
df
```

```
##                    auc        er
## Logit        0.7372900 0.3065903
## LDA          0.6381859 0.2865330
## QDA          0.6061404 0.3266476
## Naive Bayes  0.6414894 0.3094556
## KNN w/ k = 1 0.5826054 0.3553009
## k=2          0.5807391 0.3638968
## k=3          0.5779582 0.3524355
## k=4          0.5931131 0.3381089
## k=5          0.6080067 0.3180516
## k=6          0.5948488 0.3266476
## k=7          0.5859462 0.3295129
## k=8          0.6057484 0.3180516
## k=9          0.5835573 0.3266476
```

```
## k=10          0.5878126 0.3209169
```