STATS415hw4

Yunguo Cai

1.

```r
x = c(-4,-1,0,1,-1,2,3,4,7)
y = c(-1,-1,-1,-1,1,1,1,1,1)
pi = c()
mu = c()
Var = c()
x1= x[y==-1]
pi[1] = length(x1)/length(x)
mu[1] = mean(x1)
Var[1] = var(x1)
x2= x[y==1]
pi[2] = length(x2)/length(x)
mu[2] = mean(x2)
Var[2] = var(x2)
Var_common = ((length(x1)-1)*Var[1]+(length(x2)-1)*Var[2])/(length(x)-2)
```

The training data with the predictor x can be classified as categorial by defining those with y = -1 as class0, and those with y = 1 as class1. For the LDA classifiers, prior class probabilities π_0,π_1, class means μ_0,μ_1 and the pooled variance σ^2 are needed. For the QDA classifers, prior class probabilities π_0,π_1, class means μ_0,μ_1 and the class variances σ_0^2,σ_1^2 are needed. The estimated values from the training data are $\pi_0 = 0.444$,$\pi_1 = 0.556$,$\mu_0 = -1$,$\mu_1 = 3$,$\sigma^2 = 6.857$,$\sigma_0^2 = 4.667$ and $\sigma_1^2 = 8.5$.

2.

```r
mu/Var_common
```

```
## [1] -0.1458333  0.4375000
```

```r
-mu^2/Var_common/2+log(pi)
```

```
## [1] -0.8838469 -1.2440367
```

For LDA, the discriminant function for p = 1 K > 2 is

$$\delta_k(x) = x\frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + log(\pi_k)$$

The discriminant function for class 0 is

$$\delta_0(x) = -0.146x - 0.884$$

The discriminant function for class 0 is

$$\delta_1(x) = 0.438x - 1.244$$

When $\delta_0(x) > \delta_1(x)$, then x < 0.616, we assign the value of class variable y as -1 (class 0).
When $\delta_0(x) < \delta_1(x)$, then x > 0.616, we assign the value of class variable y as 1 (class 1).

```r
-log(sqrt(Var))+log(pi)
```

```
## [1] -1.581153 -1.657820
```

For QDA, the discriminant function for p = 1 is

$$\delta_k(x) = -\log(\sigma_k) - \frac{(x-\mu_k)^2}{2\sigma_k^2} + \log(\pi_k)$$

The discriminant function for class 0 is

$$\delta_0(x) = -\frac{(x+1)^2}{9.334} - 1.581$$

The discriminant function for class 0 is

$$\delta_1(x) = -\frac{(x-3)^2}{17} - 1.658$$

When $\delta_0(x) > \delta_1(x)$, then -12.543 < x < 0.811, we assign the value of class variable y as -1 (class 0). When $\delta_0(x) < \delta_1(x)$, then x < 12.543 or x > 0.811, we assign the value of class variable y as 1 (class 1).

(c)

```
lda_pred = c()
qda_pred = c()
for (i in 1:length(x)){
  if (x[i] < 0.616)
    lda_pred[i] = -1
  else
    lda_pred[i] = 1
  if (x[i] > -12.543 && x[i] < 0.811)
    qda_pred[i] = -1
  else
    qda_pred[i] = 1
}
lda_train_error = mean(y!=lda_pred)
qda_train_error = mean(y!=qda_pred)
lda_train_error
```

```
## [1] 0.2222222
```

```
qda_train_error
```

```
## [1] 0.2222222
```

The training errors for LDA and QDA are both 0.222.

(d)

```
new_x = c(-1.5, -1, 0, 1, 0.5, 1, 2.5, 5)
new_y = c(-1, -1, -1, -1, 1, 1, 1, 1)
lda_pred = c()
qda_pred = c()
for (i in 1:length(new_x)){
  if (new_x[i] < 0.616)
    lda_pred[i] = -1
  else
    lda_pred[i] = 1
  if (new_x[i] > -12.543 && new_x[i] < 0.811)
    qda_pred[i] = -1
  else
    qda_pred[i] = 1
}
lda_test_error = mean(new_y!=lda_pred)
qda_test_error = mean(new_y!=qda_pred)
lda_test_error
```

```
## [1] 0.25
```

```
qda_test_error
```

```
## [1] 0.25
```

The training errors for LDA and QDA are both 0.25.

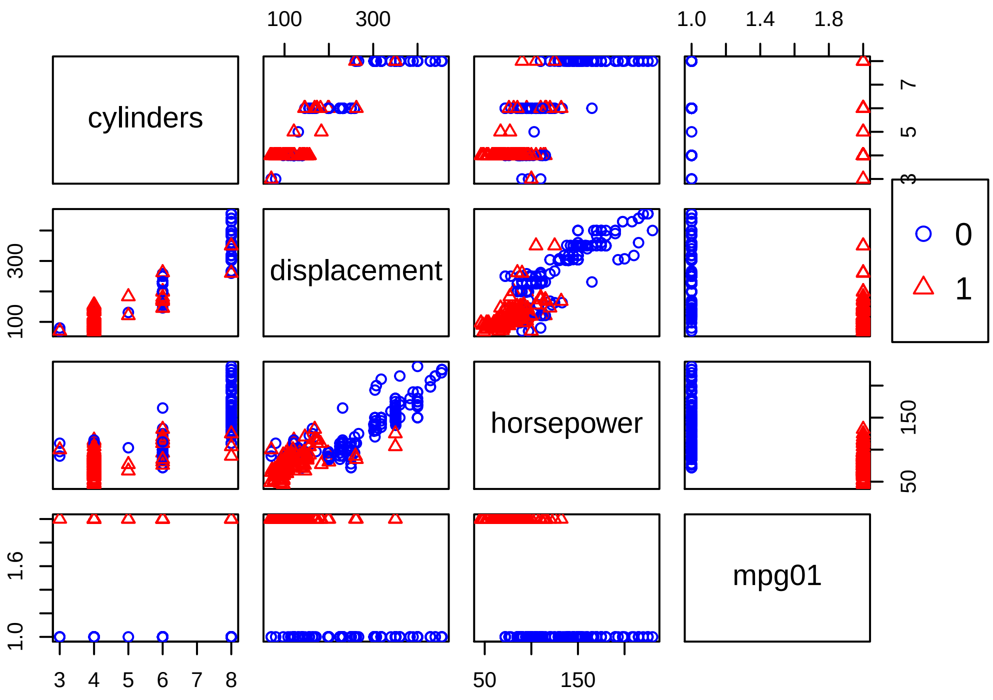
(e) LDA is more suitable for this data set. From the same training and test errors, it shows that QDA doesn't perform better than LDA in prediction though the estimated class variances are not close. LDA model is simpler with fewer parameters, so it's more suitable.

2.(a)

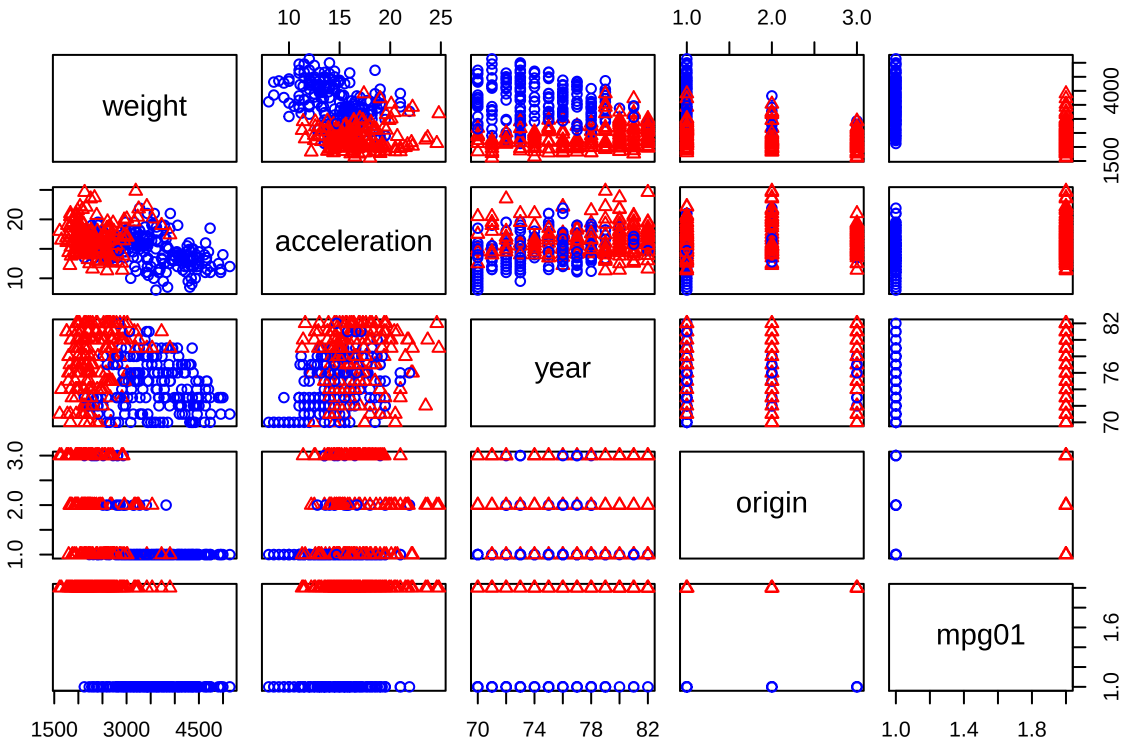
```
library(ISLR)
data("Auto")
mpg01 = rep(0,length(Auto$mpg))
median_mpg = median(Auto$mpg)
for (i in 1:length(Auto$mpg)){
  if (Auto$mpg[i] > median_mpg)
    mpg01[i] = 1
}
mpg01=as.factor(mpg01)
auto_data = data.frame(Auto,mpg01)
```

(b)

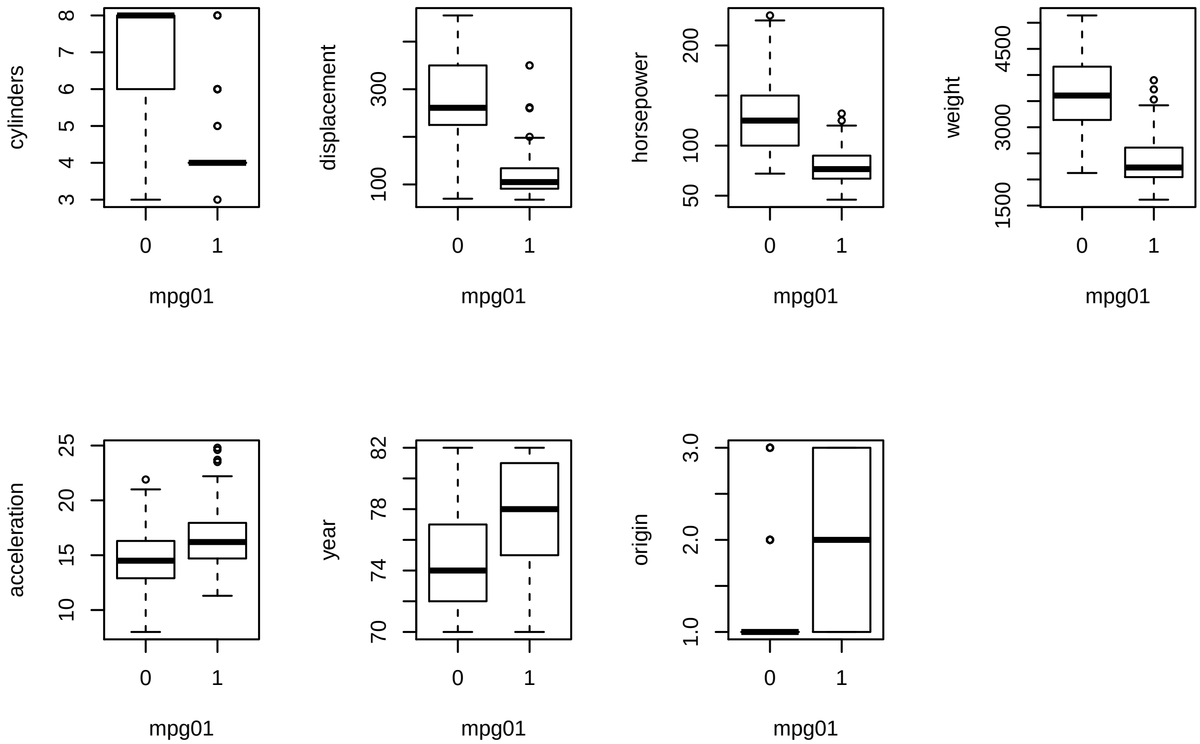
```
pairs(auto_data[c(2:4,10)], col=c("blue", "red")[auto_data$mpg01], oma=c(4,4,6,12), pch=c(1,2)[auto_dat
par(xpd=TRUE)
legend(0.8, 0.7, as.vector(unique(auto_data$mpg01)), col=c("blue", "red"), pch=1:2)
```



```r
pairs(auto_data[c(5:8,10)], col=c("blue", "red")[auto_data$mpg01], pch=c(1,2)[auto_data$mpg01])
```




```r
par(mfrow=c(2,4))
boxplot(cylinders ~ mpg01, data = auto_data, xlab = 'mpg01', ylab = 'cylinders')
boxplot(displacement ~ mpg01, data = auto_data, xlab = 'mpg01', ylab = 'displacement')
boxplot(horsepower ~ mpg01, data = auto_data, xlab = 'mpg01',
ylab = 'horsepower')
boxplot(weight ~ mpg01, data = auto_data, xlab = 'mpg01',
ylab = 'weight')
boxplot(acceleration ~ mpg01, data = auto_data, xlab = 'mpg01', ylab = 'acceleration')
boxplot(year ~ mpg01, data = auto_data, xlab = 'mpg01', ylab = 'year')
boxplot(origin ~ mpg01, data = auto_data, xlab = 'mpg01', ylab = 'origin')
```



From the scatterplots and boxplots, displacement seems most likely to be useful in predicting mpg01. Besides, weight, horsepower and acceleration all show difference for different values of mpg01, which indicates that they might be useful in predicting mpg01.

(c)

```
set.seed(12345)
auto_mpg0=which(auto_data$mpg01==0)
auto_mpg1=which(auto_data$mpg01==1)
train_id = c(sample(auto_mpg0,size=floor(0.80*length(auto_mpg0))),sample(auto_mpg1,size=floor(0.80*leng
auto_train = auto_data[train_id,]
auto_test = auto_data[-train_id,]
```

(d)

```
library(MASS)
auto_lda = lda(mpg01 ~ acceleration+displacement+horsepower+weight, data = auto_train)
auto_lda_train_pred = predict(auto_lda, auto_train)$class
auto_lda_test_pred = predict(auto_lda, auto_test)$class
calc_class_err = function(actual, predicted) {
mean(actual != predicted)
}
lda_train_err = calc_class_err(predicted = auto_lda_train_pred, actual = auto_train$mpg01)
lda_train_err
```

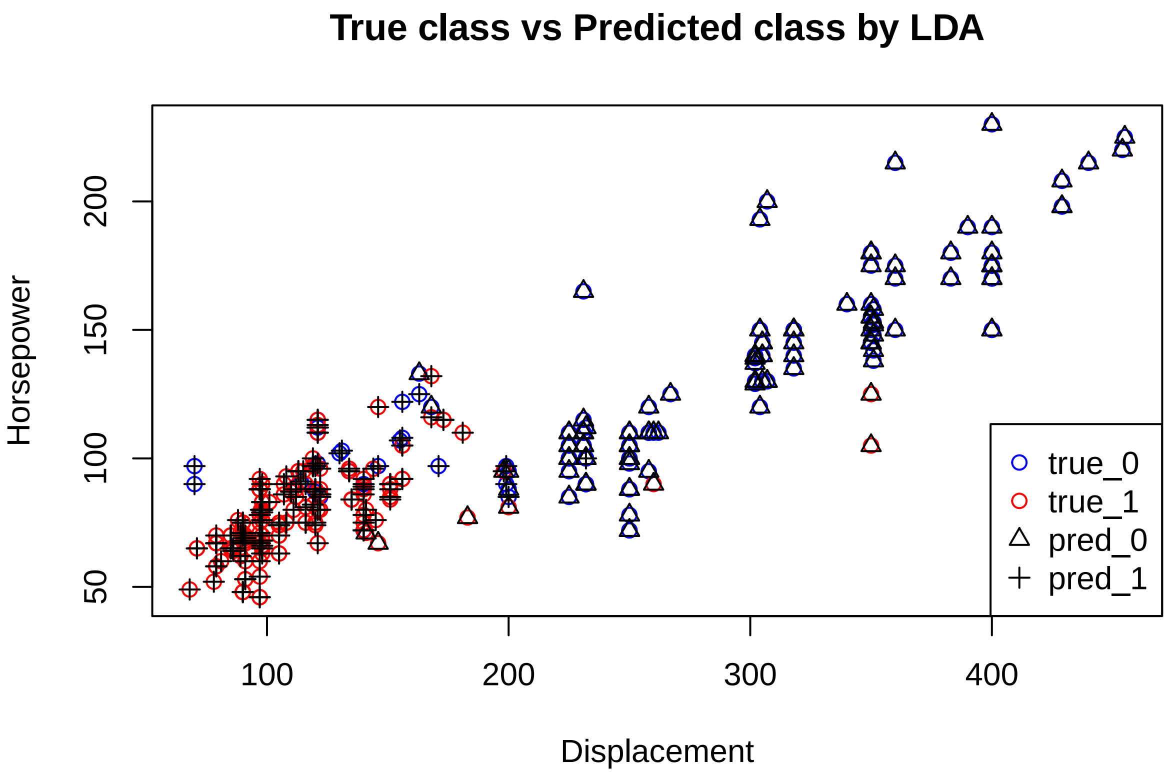
```
## [1] 0.1185897
```

```
lda_test_err = calc_class_err(predicted = auto_lda_test_pred, actual = auto_test$mpg01)
lda_test_err
```

```
## [1] 0.075
```

The LDA training error is 0.1186, and the test error is 0.075.

```r
plot(auto_train$displacement,auto_train$horsepower,
     col = c("blue", "red")[auto_train$mpg01],
     xlab = "Displacement", ylab = "Horsepower",
     main = "True class vs Predicted class by LDA"
)
points(auto_train$displacement,auto_train$horsepower,
       pch = c(2,3)[auto_lda_train_pred])
legend("bottomright", c("true_0","true_1","pred_0","pred_1"),
       col=c("blue", "red", "black", "black"),
       pch=c(1,1,2,3))
```



(e)

```r
auto_qda = qda(mpg01 ~ acceleration+displacement+horsepower+weight, data = auto_train)
auto_qda_train_pred = predict(auto_qda, auto_train)$class
auto_qda_test_pred = predict(auto_qda, auto_test)$class
calc_class_err = function(actual, predicted) {
mean(actual != predicted)
}
qda_train_err = calc_class_err(predicted = auto_qda_train_pred, actual = auto_train$mpg01)
qda_train_err
```

```
## [1] 0.1025641
```

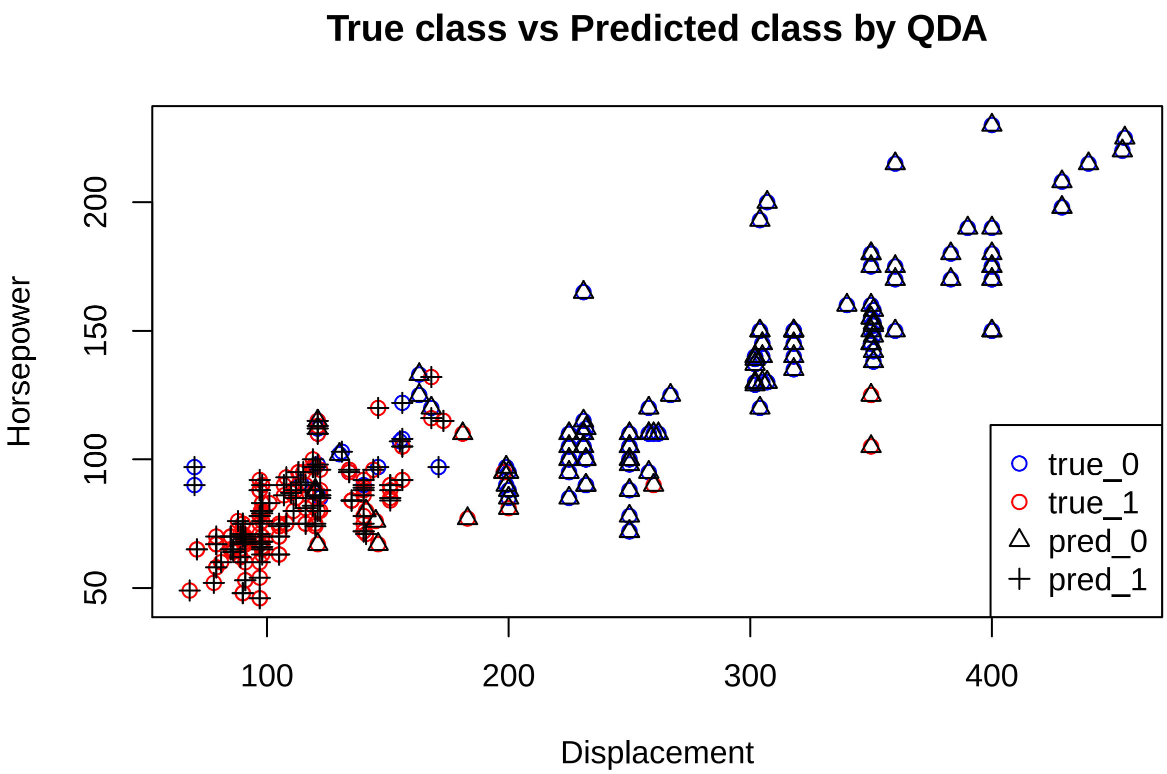
```r
qda_test_err = calc_class_err(predicted = auto_qda_test_pred, actual = auto_test$mpg01)
qda_test_err
```

```
## [1] 0.0625
```

The QDA training error is 0.1026, and the test error is 0.0625.

```r
plot(auto_train$displacement,auto_train$horsepower,
     col = c("blue", "red")[auto_train$mpg01],
```

```r
     xlab = "Displacement", ylab = "Horsepower",
     main = "True class vs Predicted class by QDA"
)
points(auto_train$displacement,auto_train$horsepower,
       pch = c(2,3)[auto_qda_train_pred])
legend("bottomright", c("true_0","true_1","pred_0","pred_1"),
       col=c("blue", "red", "black", "black"),
       pch=c(1,1,2,3))
```



(f) The training and test error of QDA model is slightly smaller than the LDA model.

```r
table(predicted = auto_lda_test_pred, actual = auto_test$mpg01)
```

```
##          actual
## predicted  0  1
##         0 35  1
##         1  5 39
```

```r
table(predicted = auto_qda_test_pred, actual = auto_test$mpg01)
```

```
##          actual
## predicted  0  1
##         0 36  1
##         1  4 39
```

From the table, the QDA model has one more correct prediction than the LDA model. If we draw the plots of the test data points, the one more correct prediction appears near the boundary, which means that the QDA has a better preformance than LDA near the boundary. This suggests that the class specific covariances are different. However, the difference in performance is not significant, so the class specific covariances are similar.