Practical 02: LDA and QDA

Alex Mounsey

1/20/2021

Linear Discriminant Analysis (LDA)

Consider a simple case with only two classes *(categories)* and two predictors. The goal is to determine a **line** which best separates the classes.

The LDA approach is based on the following procedure:

- Find the direction, represented at vector ã, such that after projecting the data perpendicularly onto this direction, the class mean values are separated by the furthest possible distance
- then, the class boundary is the line perpendicular to ã that crosses the middle point between the projected mean values

When considering the distance between the projected mean values, the spread *(standard deviation)* of projected data is taken into account.

Notes

- LDA assumes that the variability of data in the two classes is the same. However, the method often works well even if this assumption is not met
- We assumed that there are only two predictors, x_1 and x_2, but the method extends in a straightforward way to any number of p predictors: ã is then a p-dimensional vector and not a line on a two-dimensional plane
- The vector ã is called the *(first)* **canonical vector**
- The data projected orthogonally onto the direction ã is called the *(first)* **canonical variable**

KNN vs. LDA

- KNN classifiers are more flexible than LDA
- K has to be chosen, which is not always easy
- LDA classifiers are simpler than KNN, so are easier to interpret
- **For data where linear class boundaries are appropriate, LDA is the preferred method**
- **For data with more complicated class boundaries, KNN is the preferred method**

Quadtratic Discriminant Analysis (QDA)

Assume that we have g classes, and predictors in each class are distributed according to a p-dimensional normal distribution with **different** variance-covariance matrices $\sum_1, \sum_2, ..., \sum_g$ in classes.

Suppose that we construct a classifier such that for a test observation x_0:

- We first estimate the probabilities of x_0 belonging to each class respectively based on the normal distribution:

$$P(Y = 1|x_0), P(Y = 2|x_0), ..., P(Y = g|x_0)$$

- ... we then assign x_0 the class with the highest probability.

Theorem

Given the above assumptions, the classifier obtained using the above method yields quadratic class boundaries, and is called a QDA classifier.

Task A

1. Read the information about the dataset by using the command `?Auto`

```
?Auto
```

Task B

1. Create a binary variable, `mpg_01` which contains a 1 if `mpg` is above its median and a 0 otherwise

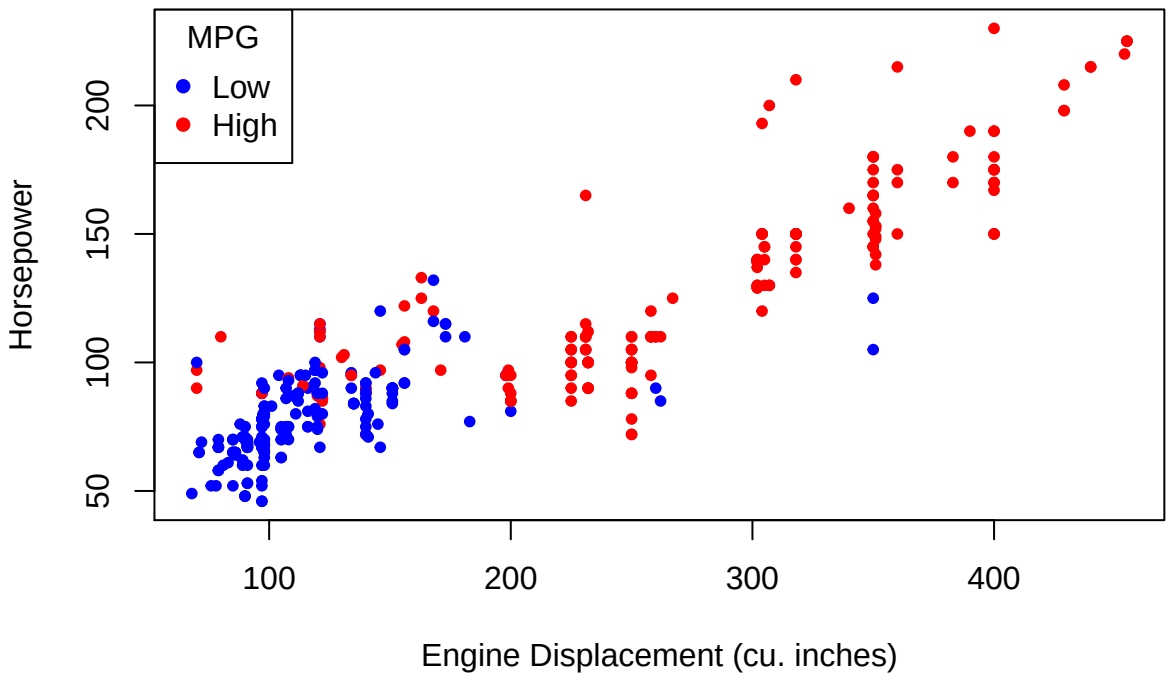
```
df <- Auto
df$mpg_01 <- as.numeric(df$mpg > median(df$mpg))
```

Task C

1. Produce a scatterplot of `horsepower` against `displacement`
    - Use colour-coding to distinguish between high and low gas mileage

```
df.colours <- rep('blue', 392)  # define a vector of colours
df.colours[df$mpg < median(df$mpg)] <- 'red'  # change colour to red when mpg is high

plot(df$displacement, df$horsepower, col = df.colours, pch = 20,
     xlab = "Engine Displacement (cu. inches)", ylab = "Horsepower")
legend('topleft', legend = c('Low', 'High'), col = c('blue', 'red'), pch = 16,
       title = "MPG")
```



Task D

1. Perform LDA in order to predict `mpg_01` based on `displacement` and `horsepower`
2. Visualise the obtained classification rule in a scatterplot of `horsepower` against `displacement`
3. Find three types of training error for this classifier:
    - The overall training error
    - The fraction of incorrectly classified cars with low mileage
    - The fraction of incorrectly classified cars with high mileage

D.1: Perform LDA

LDA can be performed using the `lda()` function, which is part of the `MASS` library. The basic general syntax is:

```
lda(y ~ x1 + x2 + ... + xp, data, subset)
```

where:

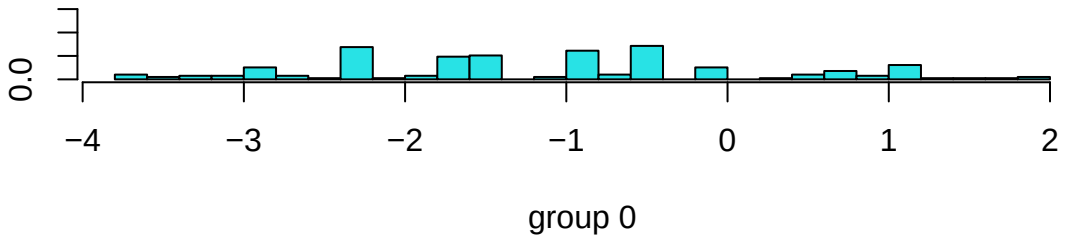
- `y` is the variable containing class labels
- `x1`, `x2`, ..., `xp` are predictors
- `data` is the dataset in which these variables are stored
- `subset` is an optional argument which holds the indices of observations from `data` to include in the analysis

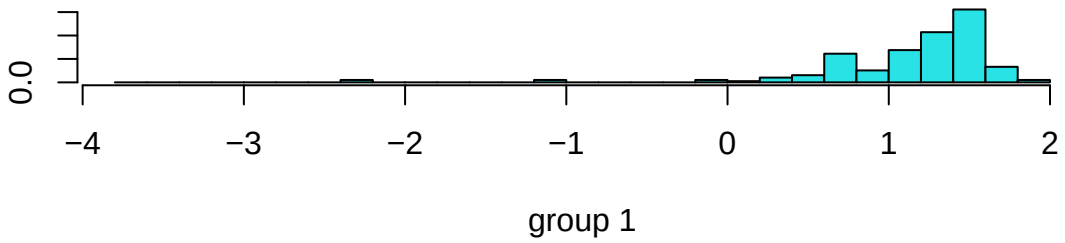
```r
df.lda <- lda(mpg_01 ~ displacement + horsepower, data = df)
df.lda
```

```
## Call:
## lda(mpg_01 ~ displacement + horsepower, data = df)
##
## Prior probabilities of groups:
##   0   1
## 0.5 0.5
##
## Group means:
##   displacement horsepower
## 0     273.1582  130.11224
## 1     115.6658   78.82653
##
## Coefficients of linear discriminants:
##                        LD1
## displacement -0.015316841
## horsepower    0.002421083
```

```r
# 'LD1' shows the coordinates of the vector that separates the two classes

plot(df.lda)
```





D.2: Visualise the Classification Rule

We can visualise the resulting classification rule by drawing the class boundary in a scatterplot of the data. As with KNN, in order to indicate colours in the class areas, we define a grid of points that cover the entire range of the data and then apply the classifier to this grid.

```
len <- 100  # grid dimensions

xp <- seq(60, 460, length = len)  # points covering range of displacement
yp <- seq(45, 260, length = len)  # points covering range of horsepower

xygrid <- expand.grid(displacement = xp, horsepower = yp)
```

Making Predictions using an LDA Classifier

In order to classify new data using an LDA classifier we use the `predict()` function. It takes two arguments: a fitted LDA object returned by the `lda()` function and new data to be classified. It returns a list consisting of three elements:

- `class`: Contains the predicted class labels based on LDA
- `posterior`: A matrix whos k^th column contains the posterior probability that the corresponding observation belongs to the k^th class
- `x`: The canonical variable *(data projected onto the canonical direction)*

```
grid.lda <- predict(df.lda, xygrid)  # classify the points from the grid

grid.lda$class[1:10]  # display teh first 10 predicted classes
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1
## Levels: 0 1
```

```
grid.lda$post[1:10, ]  # display the first 10 posterior probabilities
```

```
##             0         1
## 1  0.01235505 0.9876450
## 2  0.01420779 0.9857922
## 3  0.01633376 0.9836662
## 4  0.01877180 0.9812282
## 5  0.02156577 0.9784342
## 6  0.02476509 0.9752349
## 7  0.02842525 0.9715748
## 8  0.03260827 0.9673917
## 9  0.03738319 0.9626168
## 10 0.04282634 0.9571737
```

Creating the Plot

First, let's define the class colours:

```
grid.colours <- rep('lightblue', len*len)  # class boundary colours

# change colour to red for observations with high mileage
for (i in 1:(len*len)) if (grid.lda$class[i] == 0) grid.colours[i] <- 'mistyrose1'
```

Then we must define the quantiles $P(\texttt{mpg} = \text{low})$ - $P(\texttt{mpg} = \text{high})$. This is required in order to draw the class boundary. The class boundary is when $zp = 0$, or $P(\text{low}) = P(\text{high}) = 0.5$

```
zp <- grid.lda$post[ ,1] - grid.lda$post[ ,2]
```

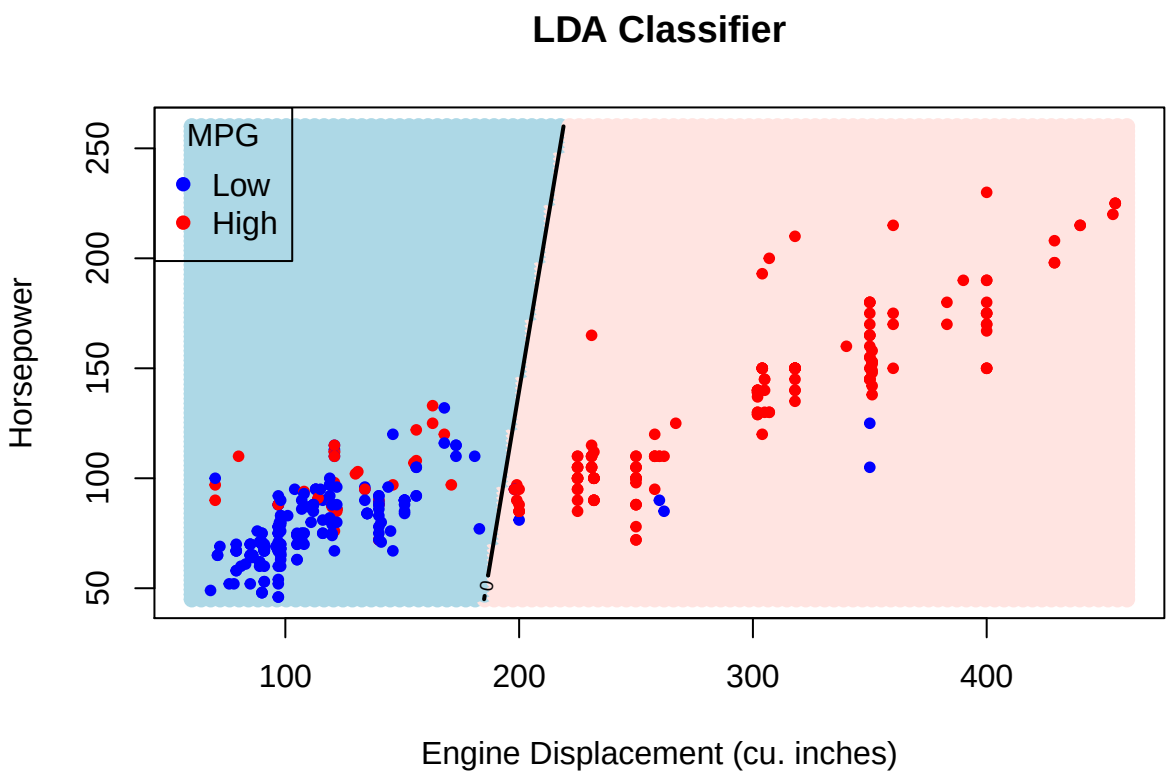
Finally, produce the plot:

```
plot(xygrid, col = grid.colours, main = "LDA Classifier",
     xlab = "Engine Displacement (cu. inches)", ylab = "Horsepower", pch = 19)

contour(xp, yp, matrix(zp, len), levels = 0, add = TRUE, lwd = 2)

points(df$displacement, df$horsepower, pch = 20, col = df.colours)

legend('topleft', legend = c('Low', 'High'), col = c('blue', 'red'), pch = 16,
       title = "MPG")
```



D.3: Training Errors

Our LDA classifier can make two types of error:

1. It can incorrectly assign an observation with low MPG to the *high MPG* category
2. It can incorrectly assign an observation with high MPG to the *low MPG* category

It is often of interest to determine which of these two types of errors are being made:

```
lda.pred <- predict(df.lda)  # predictions for the training data
lda.class <- lda.pred$class  # class predictions for the training data

lda.confm <- table(lda.class, df$mpg_01)  # confusion matrix
lda.confm
```

```
##
## lda.class   0   1
##         0 164   6
##         1  32 190
```

```
# calculate the overall fraction of incorrectly classified data
(lda.confm[1,2] + lda.confm[2,1]) / sum(lda.confm)
```

```
## [1] 0.09693878
```

```
# calculate the fraction of incorrect classifications for observations
# with low mileage
lda.confm[2,1] / (lda.confm[1,1] + lda.confm[2,1])
```

```
## [1] 0.1632653
```

```
# calculate the fraction of incorrect classifications for observations
# with high mileage
lda.confm[1,2] / (lda.confm[1,2] + lda.confm[2,2])
```

```
## [1] 0.03061224
```

Task E

1. Perform QDA in order to predict `mpg_01` based on `displacement` and `horsepower`
2. Visualise the obtained classification rule in a scatterplot of `horsepower` against 'displacement
3. Find three types of training error for this classifier:
    - The overall training error
    - The fraction of incorrectly classified cars with low mileage
    - The fraction of incorrectly classified cars with high mileage

E.1: Perform QDA

```
df.qda <- qda(mpg_01 ~ displacement + horsepower, data = df)
df.qda
```

```
## Call:
## qda(mpg_01 ~ displacement + horsepower, data = df)
##
## Prior probabilities of groups:
##   0   1
## 0.5 0.5
##
## Group means:
##   displacement horsepower
## 0     273.1582  130.11224
## 1     115.6658   78.82653
```

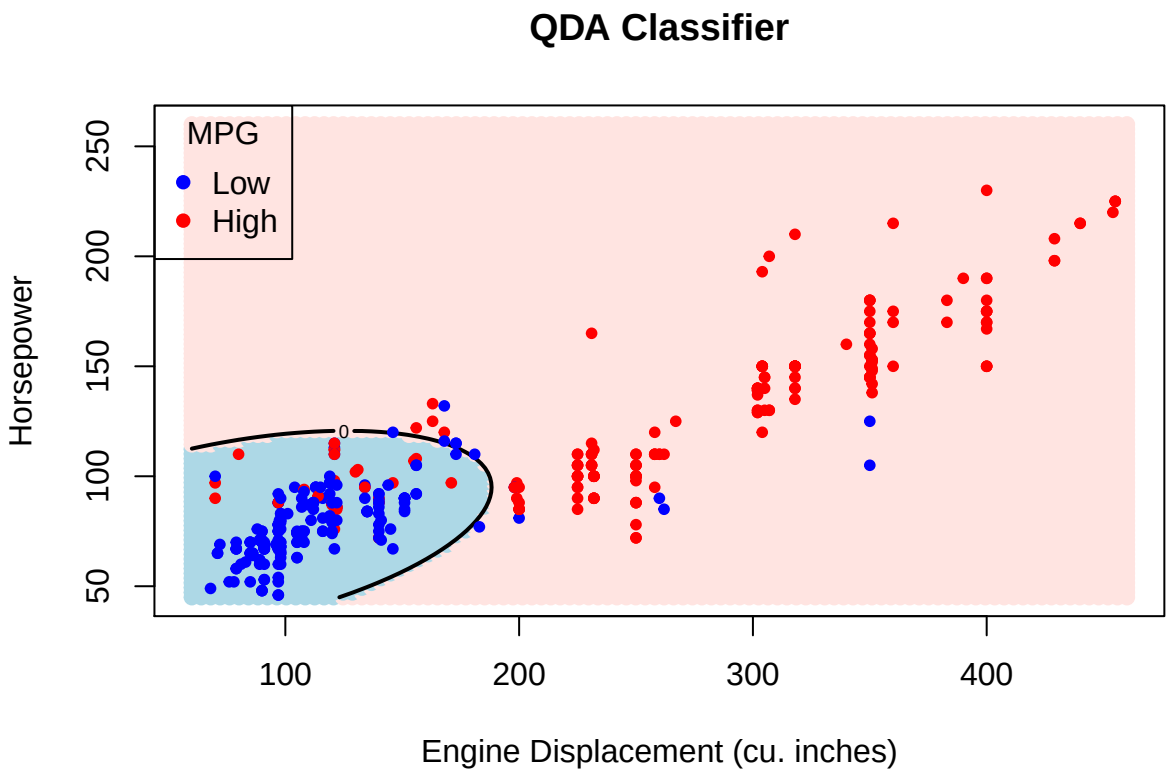
E.2: Visualise the Classification Rule

```
# classify the points from the grid
grid.qda <- predict(df.qda, xygrid)

# class boundary colours
grid.colours <- rep('lightblue', len*len)
for (i in 1:(len*len)) if (grid.qda$class[i] == 0) grid.colours[i] <- 'mistyrose1'

zp <- grid.qda$post[ ,1] - grid.qda$post[ ,2]

plot(xygrid, col = grid.colours, main = "QDA Classifier",
     xlab = "Engine Displacement (cu. inches)", ylab = "Horsepower", pch = 19)
contour(xp, yp, matrix(zp, len), levels = 0, add = TRUE, lwd = 2)
points(df$displacement, df$horsepower, pch = 20, col = df.colours)
legend('topleft', legend = c('Low', 'High'), col = c('blue', 'red'), pch = 16,
       title = "MPG")
```



E.3: Training Errors

```
qda.pred <- predict(df.qda)  # predictions for the training data
qda.class <- qda.pred$class  # class predictions for the training data

qda.confm <- table(qda.class, df$mpg_01)  # confusion matrix
qda.confm
```

```
##
## qda.class   0   1
##         0 168  13
##         1  28 183
```

```
# calculate the overall fraction of incorrectly classified data
(qda.confm[1,2] + qda.confm[2,1]) / sum(qda.confm)
```

```
## [1] 0.1045918
```

```
# calculate the fraction of incorrect classifications for observations
# with low mileage
qda.confm[2,1] / (qda.confm[1,1] + qda.confm[2,1])
```

```
## [1] 0.1428571
```

```r
# calculate the fraction of incorrect classifications for observations
# with high mileage
qda.confm[1,2] / (qda.confm[1,2] + qda.confm[2,2])
```

```
## [1] 0.06632653
```

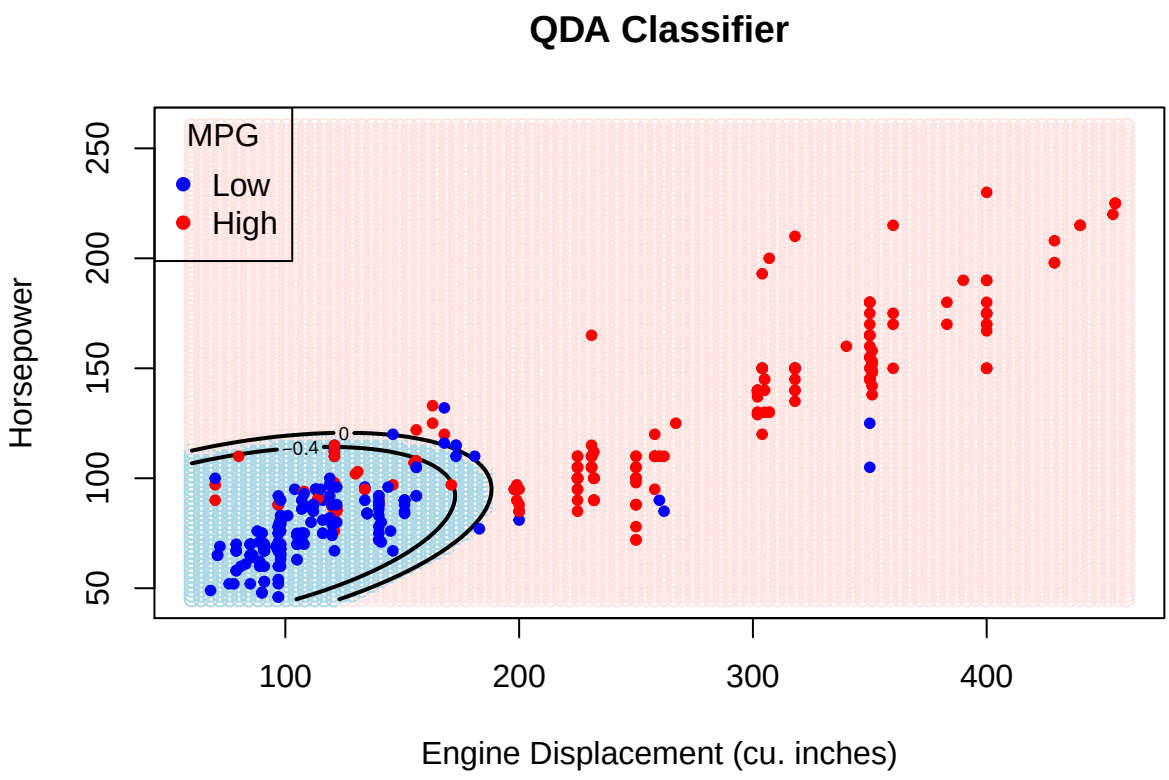
Task F

Misclassifying a car with low mileage can be potentially more costly than misclassifying a car with high mileage.

1. Visualise the QDA classifier, modified by setting the threshold for the posterior probability of low mileage to 0.3, that is an observation is classified as low mileage if $P(\text{mpg\_01} = 0) > 0.3$
2. Obtain the training errors for this modified rule

F.1: Visualise the Modified QDA Classifier

```r
plot(xygrid, col = grid.colours, main = "QDA Classifier",
     xlab = "Engine Displacement (cu. inches)", ylab = "Horsepower")
contour(xp, yp, matrix(zp, len), levels = c(0, -0.4), add = TRUE, lwd = 2)
points(df$displacement, df$horsepower, pch = 20, col = df.colours)
legend('topleft', legend = c('Low', 'High'), col = c('blue', 'red'), pch = 16,
       title = "MPG")
```



F.2: Training Errors

```
new.classes <- ifelse(qda.pred$post[,1] > 0.3, 0, 1)  # update threshold

new.confm <- table(new.classes, df$mpg_01)  # updated confusion matrix
new.confm
```

```
##
## new.classes   0   1
##           0 169  15
##           1  27 181
```

```
# calculate the overall fraction of incorrectly classified data
(new.confm[1,2] + new.confm[2,1]) / sum(new.confm)
```

```
## [1] 0.1071429
```

```
# calculate the fraction of incorrect classifications for observations
# with low mileage
new.confm[2,1] / (new.confm[1,1] + new.confm[2,1])
```

```
## [1] 0.1377551
```

```
# calculate the fraction of incorrect classifications for observations
# with high mileage
new.confm[1,2] / (new.confm[1,2] + new.confm[2,2])
```

```
## [1] 0.07653061
```

Task G

1. Compare the three classifiers based on their training errors

- LDA resulted in the lowest overall training error (9.7%)
- QDA resulted in a slightly higher overall training error (10.5%)
- QDA had a lower fraction of incorrectly classified cars with low mileage (14.3%), compared to LDA (16.3%)
- QDA had a higher fraction of incorrectly classified cars with high mileage (6.6%), compared to LDA (3.1%)
- The modified QDA had the highest overall training error (10.7%)
- The modified QDA had the lowest fraction of incorrectly classified cars with low mileage (13.8%)

If we use the fraction of incorrectly classified cars with low mileage as the priority, due to the potential costs that could be incurred, the modified QDA classifier is preferred.