Homework 2

Shrusti Ghela

4/27/2022

1. In this problem, we will make use of the Auto data set, which is part of the ISLR2 package.

```
library(ISLR2)
head(Auto)

##   mpg cylinders displacement horsepower weight acceleration year origin
## 1  18         8          307        130   3504         12.0   70      1
## 2  15         8          350        165   3693         11.5   70      1
## 3  18         8          318        150   3436         11.0   70      1
## 4  16         8          304        150   3433         12.0   70      1
## 5  17         8          302        140   3449         10.5   70      1
## 6  15         8          429        198   4341         10.0   70      1
##                        name
## 1 chevrolet chevelle malibu
## 2         buick skylark 320
## 3        plymouth satellite
## 4             amc rebel sst
## 5               ford torino
## 6          ford galaxie 500
```

a. Fit a least squares linear model to the data, in order to predict mpg using all of the other predictors except for name. Present your results in the form of a table. Be sure to indicate clearly how any qualitative variables should be interpreted.

```
Auto_1 <- Auto[1:8]
head(Auto_1)

##   mpg cylinders displacement horsepower weight acceleration year origin
## 1  18         8          307        130   3504         12.0   70      1
## 2  15         8          350        165   3693         11.5   70      1
## 3  18         8          318        150   3436         11.0   70      1
## 4  16         8          304        150   3433         12.0   70      1
## 5  17         8          302        140   3449         10.5   70      1
## 6  15         8          429        198   4341         10.0   70      1

lm.fit <- lm(mpg ~ ., data=Auto_1)
summary(lm.fit)

##
## Call:
## lm(formula = mpg ~ ., data = Auto_1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.5903 -2.1565 -0.1169  1.8690 13.0604
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -17.218435   4.644294  -3.707  0.00024 ***
## cylinders     -0.493376   0.323282  -1.526  0.12780
## displacement   0.019896   0.007515   2.647  0.00844 **
## horsepower    -0.016951   0.013787  -1.230  0.21963
## weight        -0.006474   0.000652  -9.929  < 2e-16 ***
## acceleration   0.080576   0.098845   0.815  0.41548
## year           0.750773   0.050973  14.729  < 2e-16 ***
## origin         1.426141   0.278136   5.127 4.67e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.328 on 384 degrees of freedom
## Multiple R-squared:  0.8215, Adjusted R-squared:  0.8182
## F-statistic: 252.4 on 7 and 384 DF,  p-value: < 2.2e-16
```

By data exploration, we understand the ‘origin’ is a categorical variable. Also, the ‘cylinders’ takes only integral values, so they could be treated either ways. But we have only 5 types of cylinders in the entire data, them being 3,4,5,6,8. So, we could also consider cylinders as a categorical values.

Here, the categorical variables ‘origin’ and ‘cylinders’ have numerical values. Because of this, R doesn’t consider origin and cylinders as categorical variables and instead considers them to be integral/numerical/quantitative variables. Here, the coefficient of the categorical variables represent unit change in the variable when everything else is kept constant. This is nothing but the change between categories. However, there is a problem with this. Any categorical variables can’t be fit in a regression model in it’s raw form.

One way to deal with this is to use as.factor(). This helps to convert numeric data to factor. Here, R by default takes the value of origin=1 as the baseline reference against which other categories are compared. Similarly for cylinders it takes 3 as the baseline class against which the other categories are compared.

```
lm.fit.1 <- lm(mpg ~ . - origin + as.factor(origin) - cylinders + as.factor(c
ylinders), data=Auto_1)
summary(lm.fit.1)

##
## Call:
## lm(formula = mpg ~ . - origin + as.factor(origin) - cylinders +
##     as.factor(cylinders), data = Auto_1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.6797 -1.9373 -0.0678  1.6711 12.7756
##
## Coefficients:
```

```
##                          Estimate Std. Error t value Pr(>|t|)    
## (Intercept)            -2.208e+01  4.541e+00  -4.862 1.70e-06 ***
## displacement            1.870e-02  7.222e-03   2.590  0.00997 ** 
## horsepower             -3.490e-02  1.323e-02  -2.639  0.00866 ** 
## weight                 -5.780e-03  6.315e-04  -9.154  < 2e-16 ***
## acceleration            2.598e-02  9.304e-02   0.279  0.78021    
## year                    7.370e-01  4.892e-02  15.064  < 2e-16 ***
## as.factor(origin)2      1.764e+00  5.513e-01   3.200  0.00149 ** 
## as.factor(origin)3      2.617e+00  5.272e-01   4.964 1.04e-06 ***
## as.factor(cylinders)4  6.722e+00  1.654e+00   4.064 5.85e-05 ***
## as.factor(cylinders)5  7.078e+00  2.516e+00   2.813  0.00516 ** 
## as.factor(cylinders)6  3.351e+00  1.824e+00   1.837  0.06701 .  
## as.factor(cylinders)8  5.099e+00  2.109e+00   2.418  0.01607 *  
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 3.098 on 380 degrees of freedom
## Multiple R-squared:  0.8469, Adjusted R-squared:  0.8425 
## F-statistic: 191.1 on 11 and 380 DF,  p-value: < 2.2e-16
```

Here, $\hat{\beta}_0$ represents the mean of American origin

$\hat{\beta}_{European}$ represents the difference between the mean of American origin and European origin

$\hat{\beta}_{Japanese}$ represents the difference between the mean of American origin and Japanese origin

Similarly for cylinders, $\hat{\beta}_0$ represents the mean of cars with 3 cylinders

$\hat{\beta}_{cylinders=4}$ represents the mean of cars with 3 cylinders and 4 cylinders

$\hat{\beta}_{cylinders=5}$ represents the mean of cars with 3 cylinders and 5 cylinders

$\hat{\beta}_{cylinders=6}$ represents the mean of cars with 3 cylinders and 6 cylinders

$\hat{\beta}_{cylinders=8}$ represents the mean of cars with 3 cylinders and 8 cylinders

b. What is the (training set) mean squared error of this model?

For the first model, without specifying categorical variables.

```
mean(lm.fit$residuals^2)

## [1] 10.84748
```

For the model with explicitly specifying categorical variables.

```
mean(lm.fit.1$residuals^2)

## [1] 9.301819
```

c. What gas mileage do you predict for a Japanese car with three cylinders, displacement 100, horsepower of 85, weight of 3000, acceleration of 20, built in the year 1980?

```
test_data_1c <- data.frame(matrix(c(3, 100, 85, 3000, 20, 80, 3), nrow=1))
colnames(test_data_1c)<- c(colnames(Auto_1[2:8]))
test_data_1c

##   cylinders displacement horsepower weight acceleration year origin
## 1         3          100         85   3000           20   80      3
```

For the first model (lm.fit),

```
predict(lm.fit, test_data_1c)

##        1
## 28.37978
```

For the corrected model (lm.fit.1),

```
predict(lm.fit.1, test_data_1c)

##        1
## 21.57648
```

d. On average, holding all other covariates fixed, what is the difference between the mpg of a Japanese car and the mpg of an American car?

According to model 1 (lm.fit), On average, holding all other covariates fixed, the difference between the mpg of a Japanese car and the mpg of an American car is 2*1.43 = 2.86

According to model 2 (lm.fit.1), On average, holding all other covariates fixed, the difference between the mpg of a Japanese car and the mpg of an American car is 2.62 (This is our our interpretation of the $\hat{\beta}$ for categorical variables as discussed above)

e. On average, holding all other covariates fixed, what is the change in mpg associated with a 10-unit change in horsepower?

According to lm.fit, the change in mpg associated with a 10-unit change in horsepower holding all other covariates fixed is 0.16951.

And according to lm.fit.1, the change in mpg associated with a 10-unit change in horsepower holding all other covariates fixed is 0.3490.

2. Consider using only the origin variable to predict mpg on the Auto data set. In this problem, we will explore the coding of this qualitative variable.

a. First, code the origin variable using two dummy (indicator) variables, with Japanese as the default value. Write out an equation like (3.30) in the textbook, and report the coefficient estimates. What is the predicted mpg for a Japanese car? for an American car? for a European car?

```
american <- ifelse(Auto$origin == 1,1,0)
european <- ifelse(Auto$origin == 2,1,0)
```

```
Auto_2a <- data.frame(mpg=Auto$mpg, american, european)

lm.fit.2a <- lm(mpg ~ ., data=Auto_2a)
summary(lm.fit.2a)

##
## Call:
## lm(formula = mpg ~ ., data = Auto_2a)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -12.451  -5.034  -1.034   3.649  18.966
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  30.4506     0.7196  42.314  < 2e-16 ***
## american    -10.4172     0.8276 -12.588  < 2e-16 ***
## european     -2.8477     1.0581  -2.691  0.00742 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.396 on 389 degrees of freedom
## Multiple R-squared:  0.3318, Adjusted R-squared:  0.3284
## F-statistic:  96.6 on 2 and 389 DF,  p-value: < 2.2e-16
```

$\hat{y}_i = \beta_0 = 30.4506$ if the i^{th} car is Japanese

$\hat{y}_i = \beta_0 + \beta_{American} = 30.4506 - 10.4172 = 20.0334$ if the i^{th} car is American

$\hat{y}_i = \beta_0 + \beta_{European} = 30.4506 - 2.8477 = 27.6029$ if the i^{th} car is European

Here, $\hat{y}_i$ represents the predicted value of mpg.

b. Now, code the origin variable using two dummy (indicator) variables, with American as the default. Write out an equation like (3.30) in the textbook, and report the coefficient estimates. What is the predicted mpg for a Japanese car? for an American car? for a European car?

```
japanese <- ifelse(Auto$origin == 3,1,0)
european <- ifelse(Auto$origin == 2,1,0)

Auto_2b <- data.frame(mpg=Auto$mpg, japanese , european)
head(Auto_2b)

##   mpg japanese european
## 1  18        0        0
## 2  15        0        0
## 3  18        0        0
## 4  16        0        0
## 5  17         0        0
## 6  15         0        0

lm.fit.2b <- lm(mpg ~ ., data=Auto_2b)
summary(lm.fit.2b)

##
## Call:
## lm(formula = mpg ~ ., data = Auto_2b)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -12.451  -5.034  -1.034   3.649  18.966
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  20.0335     0.4086  49.025   <2e-16 ***
## japanese     10.4172     0.8276  12.588   <2e-16 ***
## european      7.5695     0.8767   8.634   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.396 on 389 degrees of freedom
## Multiple R-squared:  0.3318, Adjusted R-squared:  0.3284
## F-statistic:  96.6 on 2 and 389 DF,  p-value: < 2.2e-16
```

$\hat{y}_i = \beta_0 = 20.0335$ if the i^{th} car is American

$\hat{y}_i = \beta_0 + \beta_{Japanese} = 20.0335 + 10.4172 = 30.4507$ if the i^{th} car is Japanese

$\hat{y}_i = \beta_0 + \beta_{European} = 20.0335 + 7.5695 = 27.603$ if the i^{th} car is European

Here, $\hat{y}_i$ represents the predicted value of mpg.

c. Now, code the origin variable using two variables that take on values of +1 or −1. Write out an equation like (3.30) in the textbook, and report the coefficient estimates. What is the predicted mpg for a Japanese car? for an American car? for a European car?

```
cjapanese <- ifelse(Auto$origin == 3,1,-1)
camerican <- ifelse(Auto$origin == 1,1,-1)

Auto_2c <- data.frame(mpg=Auto$mpg, camerican, cjapanese)
head(Auto_2c)

##   mpg camerican cjapanese
## 1  18         1        -1
## 2  15         1        -1
## 3  18         1        -1
## 4  16         1        -1
## 5  17         1        -1
## 6  15         1        -1
```

```
lm.fit.2c <- lm(mpg ~ ., data=Auto_2c)
summary(lm.fit.2c)
```

```
##
## Call:
## lm(formula = mpg ~ ., data = Auto_2c)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -12.451  -5.034  -1.034   3.649  18.966
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  25.2421     0.4138  61.003  < 2e-16 ***
## camerican    -3.7847     0.4384  -8.634  < 2e-16 ***
## cjapanese     1.4238     0.5290   2.691  0.00742 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.396 on 389 degrees of freedom
## Multiple R-squared:  0.3318, Adjusted R-squared:  0.3284
## F-statistic:  96.6 on 2 and 389 DF,  p-value: < 2.2e-16
```

$\hat{y}_i = \beta_0 - \beta_{Japanese} - \beta_{American} = 25.2421 - 1.4238 + 3.7847 = 27.603$ if the i^{th} car is European

$\hat{y}_i = \beta_0 + \beta_{Japanese} - \beta_{American} = 25.2421 + 1.4238 + 3.7847 = 30.4506$ if the i^{th} car is Japanese

$\hat{y}_i = \beta_0 + \beta_{American} - \beta_{Japanese} = 25.2421 - 3.7847 - 1.4238 = 20.0336$ if the i^{th} car is American

Here, $\hat{y}_i$ represents the predicted value of mpg.

d. Finally, code the origin variable using a single variable that takes on values of 0 for Japanese, 1 for American, and 2 for European. Write out an equation like (3.30) in the textbook, and report the coefficient estimates. What is the predicted mpg for a Japanese car? for an American car? for a European car?

```
Auto_2d <- data.frame(mpg=Auto$mpg, origin=Auto$origin)
Auto_2d$origin[Auto_2d$origin == 3] <- 0
```

Here, if we don't consider origin to be a categorical variable, we get a model that looks something like this

```
lm.fit.2d <- lm(mpg ~ origin, data=Auto_2d)
summary(lm.fit.2d)
```

```
##
## Call:
## lm(formula = mpg ~ origin, data = Auto_2d)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -14.394  -6.239  -1.167   5.761  22.751
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  25.2395     0.7332   34.42  < 2e-16 ***
## origin       -1.8453     0.6384   -2.89  0.00406 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.733 on 390 degrees of freedom
## Multiple R-squared:  0.02097,    Adjusted R-squared:  0.01846
## F-statistic: 8.354 on 1 and 390 DF,  p-value: 0.004063
```

$\hat{y}_i = \beta_0 = 25.2395$ if the i^{th} car is Japanese

$\hat{y}_i = \beta_0 + \beta_1 * x_{i1} = 25.2395 - 1.8453 = 23.3942$ if the i^{th} car is American

$\hat{y}_i = \beta_0 + \beta_1 * x_{i1} = 25.2395 - 2 * 1.8453 = 21.5489$ if the i^{th} car is European

Here, $\hat{y}_i$ represents the predicted value of mpg.

There is a problem with this approach. We considered Japanese to be 0, American to be 1 and European to be 2. If we consider this as numeric data, we are implicitly saying that European is two-times American. And our fitted values (predictions) depend on this arbitrary coding. We could do this in a way that Japaneses is 1, American is 2 and European is 3. Our answers would totally change.

This is the problem with treating categorical data in raw form in regression problems. This could be simply solved by considering the numeric data as factors using the as.factor().

```
lm.fit.2d.1 <- lm(mpg ~ as.factor(origin), data=Auto_2d)
summary(lm.fit.2d.1)

##
## Call:
## lm(formula = mpg ~ as.factor(origin), data = Auto_2d)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -12.451  -5.034  -1.034   3.649  18.966
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)         30.4506     0.7196  42.314  < 2e-16 ***
## as.factor(origin)1 -10.4172     0.8276 -12.588  < 2e-16 ***
## as.factor(origin)2  -2.8477     1.0581  -2.691  0.00742 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 6.396 on 389 degrees of freedom
## Multiple R-squared:  0.3318, Adjusted R-squared:  0.3284
## F-statistic:  96.6 on 2 and 389 DF,  p-value: < 2.2e-16
```

$\hat{y}_i = \beta_0 = 30.4506$ if the i^{th} car is Japanese

$\hat{y}_i = \beta_0 + \beta_{American} = 30.4506 - 10.4172 = 20.0334$ if the i^{th} car is American

$\hat{y}_i = \beta_0 + \beta_{European} = 30.4506 - 2.8477 = 27.6029$ if the i^{th} car is European

Here, $\hat{y}_i$ represents the predicted value of mpg.

e. Comment on your results in (a)-(d)

Overall, we considered five models. In (a), we constructed two dummy variables by considering Japanese as the default value. Here, we converted the raw categorical values 1,2,3 which could be mistaken as quantitative values to indicator variables for two classes (One-hot encoding). In (b), we did the same thing by changing the default class. We can see that when we changed the default class, we do not observe any change in the predicted values. This is because when we use dummy variables, we don't have an order to the classes and this could not be mistaken as quantitative values.

In (c), we don't encode the dummy variables using 0 and 1 but instead use -1 and 1. This changes our equation to predict mpg but does not change our final values. Because the coefficients for the model are calculated based on how the data is encoded. So, irrespective of the encoding type(i.e, 0, 1 or -1,1) we arrive at the same result.

In (d), we don't use dummy variables to encode the data. Instead we use one variable to encode three classes. There is a problem with this approach. We considered Japanese to be 0, American to be 1 and European to be 2. If we consider this as numeric data, we are implicitly saying that European is two-times American. And our fitted values (predictions) depend on this arbitrary coding. We could do this in a way that Japaneses is 1, American is 2 and European is 3. Our answers would totally change. Because of this, we arrive at different results as compared to (a), (b) and (c). This could be simply solved by considering the numeric data as factors using the as.factor() function. It Converts a column from numeric to factor. This is basically what we did using dummy variables.

3. Fit a model to predict mpg on the Auto dataset using origin and horsepower, as well as an interaction between origin and horsepower. Present your results, and write out an equation like (3.35) in the textbook. On average, how much does the mpg of a Japanese car change with a one-unit increase in horsepower? How about the mpg of an American car? a European car?

```
lm.fit.3 <- lm(mpg~ as.factor(origin) + horsepower + as.factor(origin)*horsep
ower, data =Auto )
summary(lm.fit.3)

##
## Call:
```

```
## lm(formula = mpg ~ as.factor(origin) + horsepower + as.factor(origin) *
##     horsepower, data = Auto)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.7415  -2.9547  -0.6389   2.3978  14.2495
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  34.476496   0.890665  38.709  < 2e-16 ***
## as.factor(origin)2           10.997230   2.396209   4.589 6.02e-06 ***
## as.factor(origin)3           14.339718   2.464293   5.819 1.24e-08 ***
## horsepower                   -0.121320   0.007095 -17.099  < 2e-16 ***
## as.factor(origin)2:horsepower -0.100515   0.027723  -3.626 0.000327 ***
## as.factor(origin)3:horsepower -0.108723   0.028980  -3.752 0.000203 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.422 on 386 degrees of freedom
## Multiple R-squared:  0.6831, Adjusted R-squared:  0.679
## F-statistic: 166.4 on 5 and 386 DF,  p-value: < 2.2e-16
```

$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_3 * x_{ihorsepower} = 34.476496 - 0.121320 * horsepower$ if the i^{th} car is American

$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 + \hat{\beta}_3 * x_{ihorsepower} + \hat{\beta}_4 * x_{ihorsepower} = 34.476496 + 10.997230 - 0.121320 * horsepower - 0.100515 * horsepower = 45.47373 - 0.221835 * horsepower$ if the i^{th} car is European

$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_2 + \hat{\beta}_3 * x_{ihorsepower} + \hat{\beta}_5 * x_{ihorsepower} = 34.476496 + 14.339718 - 0.121320 * horsepower - 0.108723 * horsepower = 48.81621 - 0.230043 * horsepower$ if the i^{th} car is Japanese

On average, for a Japanese car, with one-unit increase in horsepower, the mpg decreases by 0.230043 units.

On average, for a European car, with one-unit increase in horsepower, the mpg decreases by 0.221835 units.

On average, for a American car, with one-unit increase in horsepower, the mpg decreases by 0.121320 units.

4. Consider using least squares linear regression to predict weight (Y) using height.

a. Suppose that you measure height in inches (X1), fit the model $Y = \beta_0 + \beta_1 X_1 + \epsilon$ and obtain the coefficient estimates $\hat{\beta}_0 = -165.1$ and $\hat{\beta}_1 = 4.8$ What weight will you predict for an individual who is 64 inches tall?

```
pred <- -165.1 + 4.8*64
pred
```

```
## [1] 142.1
```

b. Now suppose that you want to measure height in feet (X2) instead of inches. (There are 12 inches to a foot.) You fit the model $Y = \beta_0^* + \beta_1^* X_2 + \epsilon$ What are the coefficient estimates? What weight will you predict for an individual who is 64 inches tall (i.e. 5.333 feet)?

$$Y = \beta_0 + \beta_1 X_1 + \epsilon = \beta_0^* + \beta_1^* X_2 + \epsilon$$

But we know,

$$X_2 * 12 = X_1$$

$$\beta_0 + \beta_1(12X_2) = \beta_0^* + \beta_1^* X_2$$

$$\beta_0 + 12\beta_1 X_2 = \beta_0^* + \beta_1^* X_2$$

$$\beta_0^* = \beta_0$$

and $\beta_1^* = 12\beta_1$

$$\beta_0^* = -165.1$$

$$\beta_1^* = 57.6$$

```
pred1 <- -165.1 + 57.6*5.333
pred1

## [1] 142.0808
```

c. Now suppose you fit the model $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon$, which contains both height in inches and height in feet as predictors. Provide a general expression for the least squares coefficient estimates for this model.

Our main intuition in Linear Regression is that the coefficients of the variables denoted by β_p is nothing but the change in the prediction or the output variable when there is a unit-change in X_p such that all other variables remain constant.

However, in this case, we can't interpret the $\beta_1 X_1$ and $\beta_2 X_2$ using that intuition. Because of the fact that $X2 = X1/12$. If X1 increases, X2 automatically increases. There is no possibility that there is a unit change in X1 and X2 remains constant. X1 and X2 are practically the same variables just on a different scale. Hence, they are perfectly correlated.

When independent variables are highly correlated, change in one variable would cause change to another and so the model results fluctuate significantly. The model results will be unstable and vary a lot given a small change in the data or model.

Looking at it mathematically, $X^T X$ would not be invertible because $X2 = X1/12$ and hence we can't calculate $\hat{\beta}$

Another way to look at it is, Let us consider the model that only depends on X1

$$Y = \beta_0 + \beta_1 X1 + \epsilon$$

It could be written as,

$$Y = \beta_0 + (\beta_1/2)X1 + (\beta_1/2)X1 + \epsilon$$

$$\Longrightarrow Y = \beta_0 + (\beta_1/2)X1 + (\beta_1/2)X1 + \epsilon$$

$$\Longrightarrow Y = \beta_0 + (\beta_1/2)X1 + \beta_1(X1/2) + \epsilon$$

$$\Longrightarrow Y = \beta_0 + (\beta_1/2)X1 + \beta_1(6X2) + \epsilon$$

$$\Longrightarrow Y = \beta_0 + (\beta_1/2)X1 + 6\beta_1 X2 + \epsilon$$

This could be re-written as,

$$Y = \beta_0 + \beta_3 X1 + \beta_2 X2 + \epsilon$$

where $\beta_3 = \beta_1/2$ and $\beta_2 = 6\beta_1$

So, these β_3 and β_2 could be essentially calculated using just X1 or X2. Hence, we verify two problems that are caused due to multicollinearity.

- The coefficient estimates can swing wildly based on which other independent variables are in the model. The coefficients become very sensitive to small changes in the model.
- Multicollinearity reduces the precision of the estimated coefficients, which weakens the statistical power of the regression model. Essentially, Multicollinearity makes it hard to interpret your coefficients, and it reduces the power of the model to identify independent variables that are statistically significant.

Dealing with this is fairly simple. One of the simplest way to deal with this is to regularize this equation. We could use Lasso to generate sparse input variables.

d. How do the (training set) mean squared errors compare for three models fit in (a)–(c)?

The training MSE for models in a and b would be equal, because they are practically the same model. However, due to multicollinearity, the model of c does not mathematically exist. Hence MSE is out of question.

If performing linear regression in R or Python, we will obtain a linear model that has MSE same as MSE of models from a and b. This basically tells us that the predictive power of the model involving multicollinearity is not affected. However, this is surely not a good model for statistical interpretation.

5. Suppose we wish to perform classification of a binary response in a setting with $p = 1: that is, X \in \mathbb{R}, and Y \in 1{,}2$. We assume that the observations in Class 1 are drawn from a $\mathcal{N}(\mu, \sigma^2)$ distribution, and that the observations in Class 2 are drawn from an Uniform[−2, 2] distribution.

a. Derive an expression for the Bayes decision boundary: that is, for the set of x such that $P(Y = 1|X = x) = P(Y = 2|X = x)$. Write it out as simply as you can.

$$P(Y = 1|X = x) = P(Y = 2|X = x)$$

Using the Bayes' theorem,

$$P(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{i=1}^{n} \pi_i f_i(x)}$$

For Bayes' Decision boundary, $\frac{\pi_1 f_1(x)}{\sum_{i=1}^{2} \pi_i f_i(x)} = \frac{\pi_2 f_2(x)}{\sum_{i=1}^{2} \pi_i f_i(x)}$

This boundary only exist in the range where both the classes are defined, in cases when only one of the class exist, that would only be the true class.

So, for the problem at hand, the range where both these will exist would be when $x \in [-2{,}2]$

$$\Rightarrow \pi_1 f_1(x) = \pi_2 f_2(x)$$

From the question we know that,

$$f_1(x) = \frac{1}{\sqrt{2\pi\sigma}} \exp(-\frac{1}{2\sigma^2}(x-\mu)^2)$$

$$f_2(x) = \frac{1}{2-(-2)} = \frac{1}{4} for - 2 \le x \le 2$$

$$(\frac{1}{\sqrt{2\pi\sigma}} \exp(-\frac{1}{2\sigma^2}(x-\mu)^2))\pi_1 = (\frac{1}{4})\pi_2$$

Rearranging the terms, we get,

$$x = \mu \pm \sqrt{2\sigma^2 (log(\frac{\sqrt{8}\pi_1}{\sqrt{\pi}(1-\pi_1)\sigma}))}$$

These values of x provide the Bayes decision boundary.

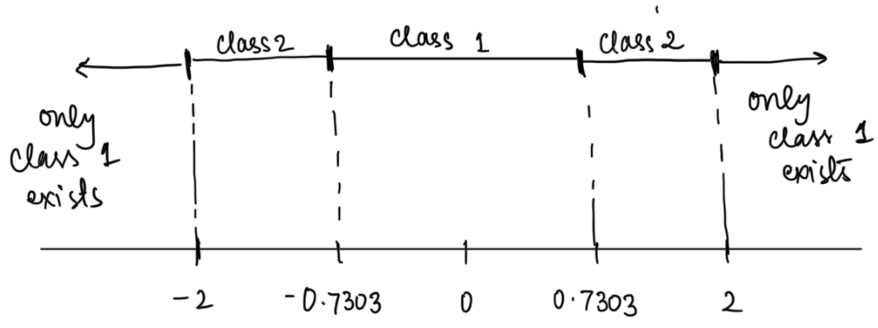
b. Suppose (for this sub-problem only) that $\mu = 0, \sigma = 1, \pi_1 = 0.45$ (here, π_1 is the prior probability that an observation belongs to Class 1). Describe the Bayes classifier in this case: what range of x values will get assigned to Class 1, and what range of x values will get assigned to Class 2? Write out your answer as simply as you can. Draw a picture showing the set of x values assigned to Class 1 and the set of x values assigned to Class 2.

Using the equation derived in the above question, we substitute the values of μ, σ, π_1 from the question and we get $x = \pm 0.73032$

So, for $x \in [-0.7303, 0.7303]$, Bayes classifier predicts class 1.

and for $x \in [-2, -0.7303] and [0.7303, 2]$, Bayes classifier predicts class 2.

for $x \geq 2$ and $x \leq 2$ there only exists one class, there is no uncertainty.



c. Now suppose we observe n training observations $(x_1, y_1), \ldots, (x_n, y_n)$. Explain how you could use these observations to estimate μ, σ, π_1 (instead of using the values that were given in part (b))

Now suppose we are given data with n training observations, we could easily estimate μ, σ, π_1

$$\hat{\mu} = \frac{1}{n_1} \sum_{i:y_i=1} x_i$$

$$\hat{\sigma} = \sqrt{\frac{\sum_{i:y_i=1} (x_i - \mu)^2}{n_1 - 1}}$$

$$\hat{\pi}_1 = \frac{n_1}{n}$$

d. Given a test observation $X = x_0$, provide an estimate of $P(Y = 1|X = x_0)$. Your answer should involve only the training observations $(x_1, y_1), \ldots, (x_n, y_n)$ and the test observation x_0, and no unknown parameters.

$$P(Y = 1|X = x_0) = \frac{\pi_1 f_1(x)}{\sum_{i=1}^{2} \pi_i\, f_i(x)}$$

$$P(Y = 1|X = x_0) = \frac{\pi_1 f_1(x)}{\pi_1 f_1(x) + \pi_2 f_2(x)}$$

$$= \frac{(\frac{n_1}{n})[\frac{1}{\sqrt{2\pi}\sqrt{\frac{\sum_{i:y_i=1}\left(x_i - \frac{1}{n_1}\sum_{i:y_i=1} x_i\right)^2}{n_1}}}\exp(-\frac{(x_0 - \frac{1}{n_1}\sum_{i:y_i=1} x_i)^2}{(2\frac{\sum_{i:y_i=1}\left(x_i - \frac{1}{n_1}\sum_{i:y_i=1} x_i\right)^2}{n_1})})]}{(\frac{n_1}{n})(\frac{1}{\sqrt{2\pi}\sqrt{\frac{\sum_{i:y_i=1}\left(x_i - \frac{1}{n_1}\sum_{i:y_i=1} x_i\right)^2}{n_1}}}\exp(-\frac{(x_0 - \frac{1}{n_1}\sum_{i:y_i=1} x_i)^2}{(2\frac{\sum_{i:y_i=1}\left(x_i - \frac{1}{n_1}\sum_{i:y_i=1} x_i\right)^2}{n_1})})) + \frac{1}{4}\frac{n - n_1}{n}}$$

6. This problem has to do with logistic regression.

a. Suppose you fit a logistic regression to some data and find that for a given observation $x = (x_1, \ldots, x_p)^T$, the estimated log-odds equals 0.7. What is $P(Y = 1|X = x)$?

$$log(\frac{p(x)}{1 - p(x)}) = 0.7$$

From Logistic Regression, we knot that $0.7 = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p$

$$P(Y = 1|X = x) = \frac{e^{\beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p}}{1 + e^{\beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p}}$$

$$= \frac{e^{0.7}}{1 + e^{0.7}}$$

$$= \frac{2.01375}{3.01375}$$

$$\Longrightarrow P(Y = 1|X = x) = 0.66819$$

b. In the same setting as (a), suppose you are now interested in the observation $x *= (x_1 + 1, x_2 - 1, x_3 + 2, x_4, \ldots, x_p)^T$. In other words, $x_1^* = x_1 + 1, x_2^* = x_2 - 1, x_3^* = x_3 + 2$, and $x_j^* = x_j for j = 4, \ldots, p$. Write out a simple expression for $P(Y = 1|X = x *)$. Your answer will involve the estimated coefficients in the logistic regression model, as well as the number 0.7.

We already know that, $0.7 = \beta_0 + \beta_1 X_1 + \ldots + \beta_p X_p$

Rewriting it in terms of X*

$$0.7 = \beta_0 + \beta_1(X_1^* - 1) + \beta_2(X_2^* + 1) + \beta_3(X_3^* - 2) \ldots + \beta_p X_p$$

$$0.7 = \beta_0 - \beta_1 + \beta_2 - 2\beta_3 + \beta_1 X_1^* + \beta_2 X_2^* + \beta_3 X_3^* \ldots + \beta_p X_p$$

$$0.7 + \beta_1 - \beta_2 + 2\beta_3 = \beta_0 + \beta_1 X_1^* + \beta_2 X_2^* + \beta_3 X_3^* \ldots + \beta_p X_p$$

$$P(Y = 1|X = x *) = \frac{e^{0.7 + \beta_1 - \beta_2 + 2\beta_3}}{1 + e^{0.7 + \beta_1 - \beta_2 + 2\beta_3}}$$

7. In this problem, you will generate data with p = 2 features and a qualitative response with K = 3 classes, and n = 50 observations per class. You will then apply linear discriminant analysis to the data.

a. Generate data such that the distribution of an observation in the kth class follows a $\mathcal{N}(\mu_k, \Sigma)$ distribution, for k = 1, . . . , K. That is, the data follow a bivariate normal distribution with a mean vector μ_k that is specific to the kth class, and a covariance matrix Σ that is shared across the K classes. Choose Σ and $\mu_1, \ldots, \mu_k$ such that there is some overlap between the K classes, i.e. no linear decision boundary is able to perfectly separate the training data. Specify your choices for Σ and $\mu_1, \ldots, \mu_k$.

$$\mu_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

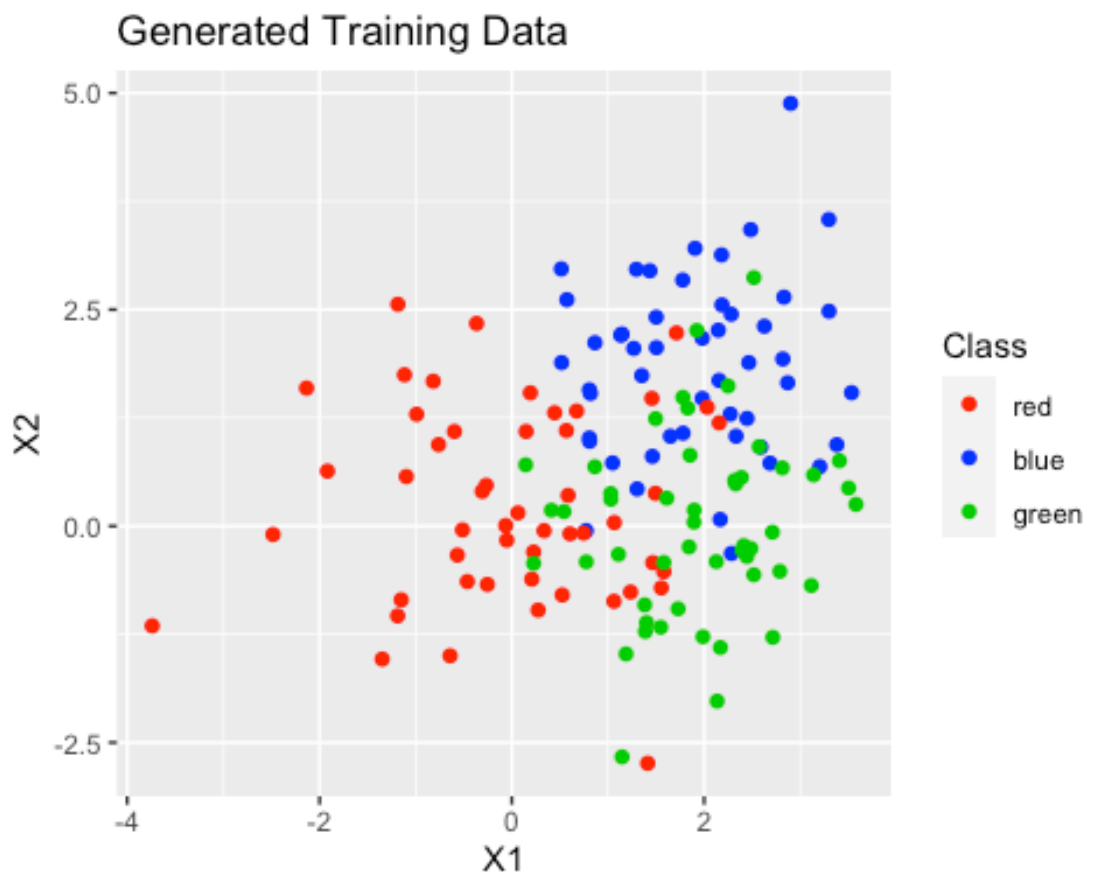
$$\mu_2 = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$$

$$\mu_3 = \begin{pmatrix} 2 \\ 0 \end{pmatrix}$$

$$\Sigma = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

```
library(MASS)

##
## Attaching package: 'MASS'

## The following object is masked from 'package:ISLR2':
##
##     Boston
```

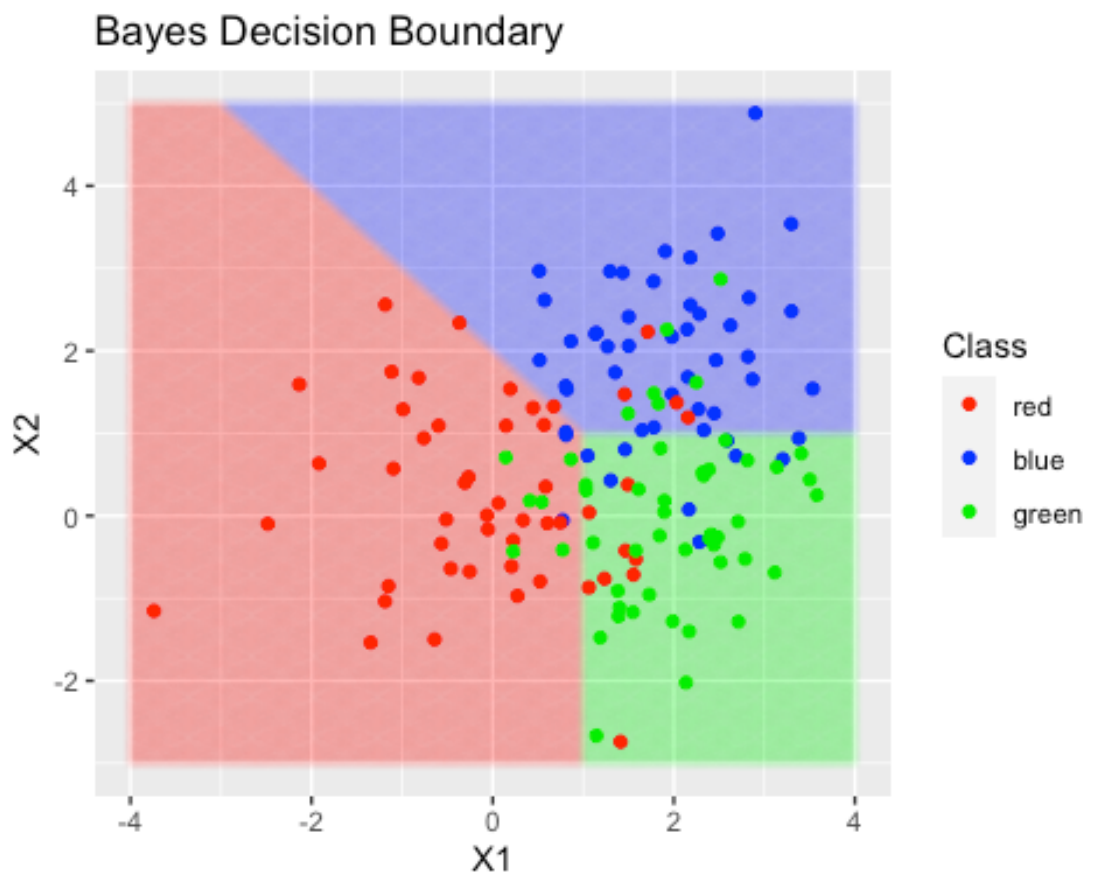
```r
library(mvtnorm)

set.seed(304)

mu_1 <- c(0,0)
mu_2 <- c(2, 2)
mu_3 <- c(2, 0)
sigma <- matrix(c(1,0,0,1), 2,2)

red_data <- mvrnorm(50, mu_1, sigma)
blue_data <- mvrnorm(50, mu_2, sigma)
green_data <- mvrnorm(50, mu_3, sigma)

trainRed7a <- data.frame(red_data, Group ="red")
trainBlue7a <- data.frame(blue_data, Group = "blue")
trainGreen7a <- data.frame(green_data, Group = "green")
trainData7a <- rbind.data.frame(trainBlue7a, trainRed7a, trainGreen7a)

colors = c("red"="red", "blue" = "blue", "green" = "green3")

library(ggplot2)

ggplot(trainData7a, aes(x=X1, y=X2))+
  geom_point(size=2, pch=16, aes(color=Group))+
  labs(x = "X1",
       y="X2",
       title = "Generated Training Data",
       color="Class")+
  scale_color_manual(values=colors)
```



b. Plot the data, with the observations in each class displayed in a different color. Compute and display the Bayes decision boundary (or Bayes decision boundaries) on this plot. This plot should look something like the right-hand panel of Figure 4.6 in the textbook (although no need to worry about shading the background, and also you don't need to display the LDA decision boundary for this sub-problem — you will do that in the next sub-problem). Be sure to label which region(s) of the plot correspond to each class.

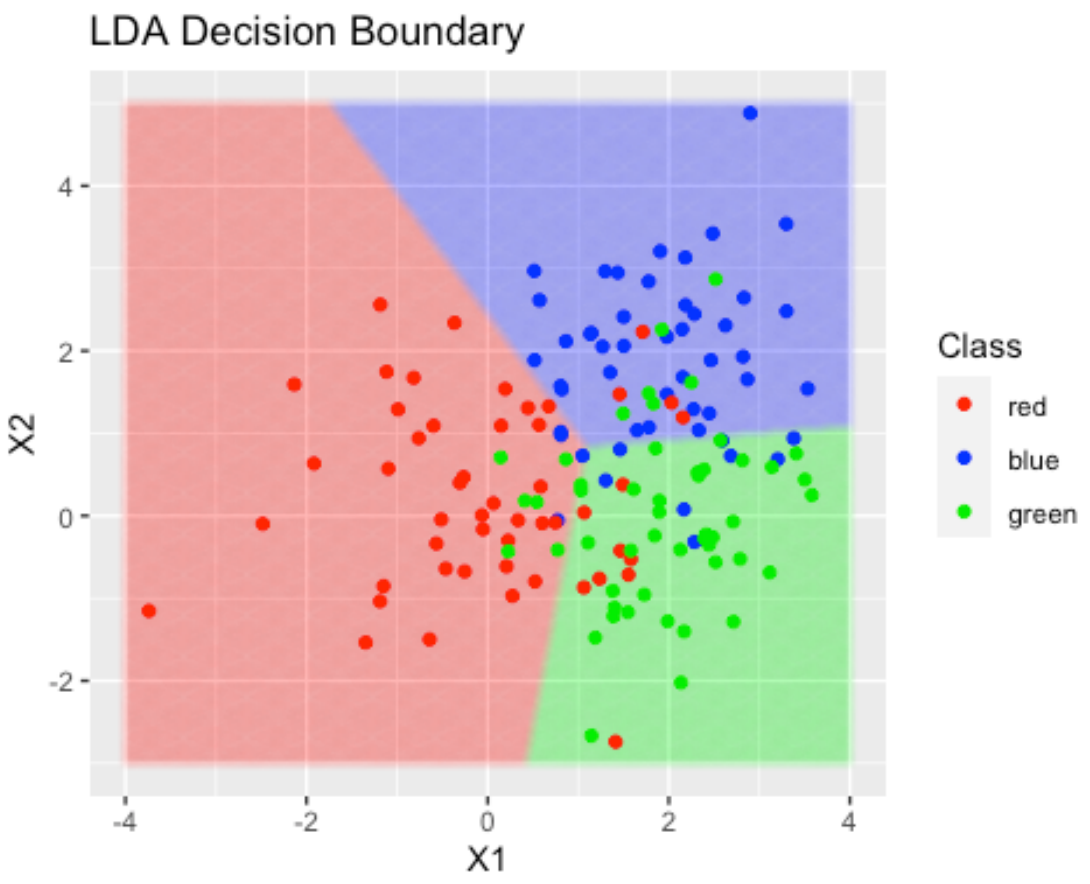
```
x <- seq(-4, 4, length.out = 300)
y <- seq(-3, 5, length.out = 300)
test_data <- expand.grid(X1 = x, X2 = y)
#head(test_data)

library(dplyr)

##
## Attaching package: 'dplyr'

## The following object is masked from 'package:MASS':
##
##     select

## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
red <- dmvnorm(test_data, mean = mu_1, sigma = sigma, log = FALSE, checkSymme
try = TRUE)
blue  <- dmvnorm(test_data, mean = mu_2, sigma = sigma, log = FALSE, checkSym
metry = TRUE)
green <- dmvnorm(test_data, mean = mu_3, sigma = sigma, log = FALSE, checkSym
metry = TRUE)

test_data <- cbind.data.frame(test_data, red, blue, green)

test_data <-test_data %>% mutate(pred =
                     case_when(red > blue & red > green ~ "red",
                               green>blue & green>red ~ "green",
                               blue>red & blue>green ~ "blue"))

colors <- c("red" = "red", "blue"="blue", "green"= "green2")

ggplot() +
geom_point(data=test_data, aes(x=X1, y=X2, color=pred), alpha=0.01)+
geom_point(data=trainData7a, aes(x=X1, y=X2, color=Group)) +
  labs(x = "X1",
       y="X2",
       title = "Bayes Decision Boundary",
       color="Class")+
  scale_color_manual(values=colors)
```



c. Fit a linear discriminant analysis model to the data, and make a plot that displays the observations as well as the decision boundary (or boundaries) corresponding to this fitted model. How does the LDA decision boundary (which can be viewed as an estimate of the Bayes decision boundary) compare to the Bayes decision boundary that you computed and plotted in (b)?

```
library(MASS)
lda.fit <- lda(Group~X1+X2, data=trainData7a)

test_data <- cbind(test_data, pred_lda = predict(lda.fit, test_data)$class)

ggplot() +
geom_point(data=test_data, aes(x=X1, y=X2, color=pred_lda), alpha=0.01)+
geom_point(data=trainData7a, aes(x=X1, y=X2, color=Group)) +
  labs(x = "X1",
       y="X2",
       title = "LDA Decision Boundary",
       color="Class")+
  scale_color_manual(values=colors)
```



We see that LDA does a pretty good job estimating the Bayes Classifier that is the optimal case.

d. Report the K × K confusion matrix for the LDA model on the training data. The rows of this confusion matrix represent the predicted class labels, and the columns represent the true class labels. (See Table 4.4 in the textbook for an example of a confusion matrix.) Also, report the training error (i.e. the proportion of training observations that are mis-classified).

```
trainData7a <- cbind(trainData7a, pred_lda = predict(lda.fit, trainData7a)$cl
ass)
head(trainData7a)

##          X1        X2 Group pred_lda
## 1 0.8067905 1.0166378  blue      red
## 2 1.6496980 1.0371858  blue     blue
## 3 0.8061965 1.5726382  blue     blue
## 4 2.1475053 2.2617550  blue     blue
## 5 2.4841975 3.4225329  blue     blue
## 6 2.1668900 0.0762792  blue    green

library(caret)

## Loading required package: lattice
```

```
expected_value <- factor(trainData7a$pred_lda)
predicted_value <- factor(trainData7a$Group)
cm7d <- confusionMatrix(data=predicted_value, reference = expected_value)

cm7d
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction blue green red
##      blue    38     8   4
##      green    6    38   6
##      red      4     8  38
##
## Overall Statistics
##
##                Accuracy : 0.76
##                  95% CI : (0.6835, 0.8259)
##     No Information Rate : 0.36
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.64
##
##  Mcnemar's Test P-Value : 0.9029
##
## Statistics by Class:
##
##                      Class: blue Class: green Class: red
## Sensitivity               0.7917       0.7037     0.7917
## Specificity               0.8824       0.8750     0.8824
## Pos Pred Value            0.7600       0.7600     0.7600
## Neg Pred Value            0.9000       0.8400     0.9000
## Prevalence                0.3200       0.3600     0.3200
## Detection Rate            0.2533       0.2533     0.2533
## Detection Prevalence      0.3333       0.3333     0.3333
## Balanced Accuracy         0.8370       0.7894     0.8370
```

Training error = 0.24

e. Generate n = 50 test observations in each of the K classes, using the bivariate normal distributions from (a). Report the K × K confusion matrix, as well as the test error, that results from applying the model fit to the training data in (c) to your test data.

```
set.seed(31)

red_D<- mvrnorm(50, mu_1, sigma)
blue_D <- mvrnorm(50, mu_2, sigma)
green_D <- mvrnorm(50, mu_3, sigma)

red_7e<- data.frame(red_D, Group ="red")
blue_7e <- data.frame(blue_D, Group = "blue")
```

```
green_7e <- data.frame(green_D, Group = "green")
test_data_7e <- rbind.data.frame(red_7e, blue_7e, green_7e)

test_data_7e <- cbind(test_data_7e, pred_lda = predict(lda.fit, test_data_7e)
$class)

expected_value7e <- factor(test_data_7e$pred_lda)
predicted_value7e <- factor(test_data_7e$Group)
cm7e <- confusionMatrix(data=predicted_value7e, reference = expected_value7e)

cm7e

## Confusion Matrix and Statistics
##
##             Reference
## Prediction blue green red
##      blue    44     2   4
##      green    7    35   8
##      red      1     6  43
##
## Overall Statistics
##
##                Accuracy : 0.8133
##                  95% CI : (0.7416, 0.8722)
##     No Information Rate : 0.3667
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.72
##
##  Mcnemar's Test P-Value : 0.1821
##
## Statistics by Class:
##
##                      Class: blue Class: green Class: red
## Sensitivity               0.8462       0.8140     0.7818
## Specificity               0.9388       0.8598     0.9263
## Pos Pred Value            0.8800       0.7000     0.8600
## Neg Pred Value            0.9200       0.9200     0.8800
## Prevalence                0.3467       0.2867     0.3667
## Detection Rate            0.2933       0.2333     0.2867
## Detection Prevalence      0.3333       0.3333     0.3333
## Balanced Accuracy         0.8925       0.8369     0.8541
```

Test Error = 0.187

f. Compare your results from (d) and (e), and comment on your findings.

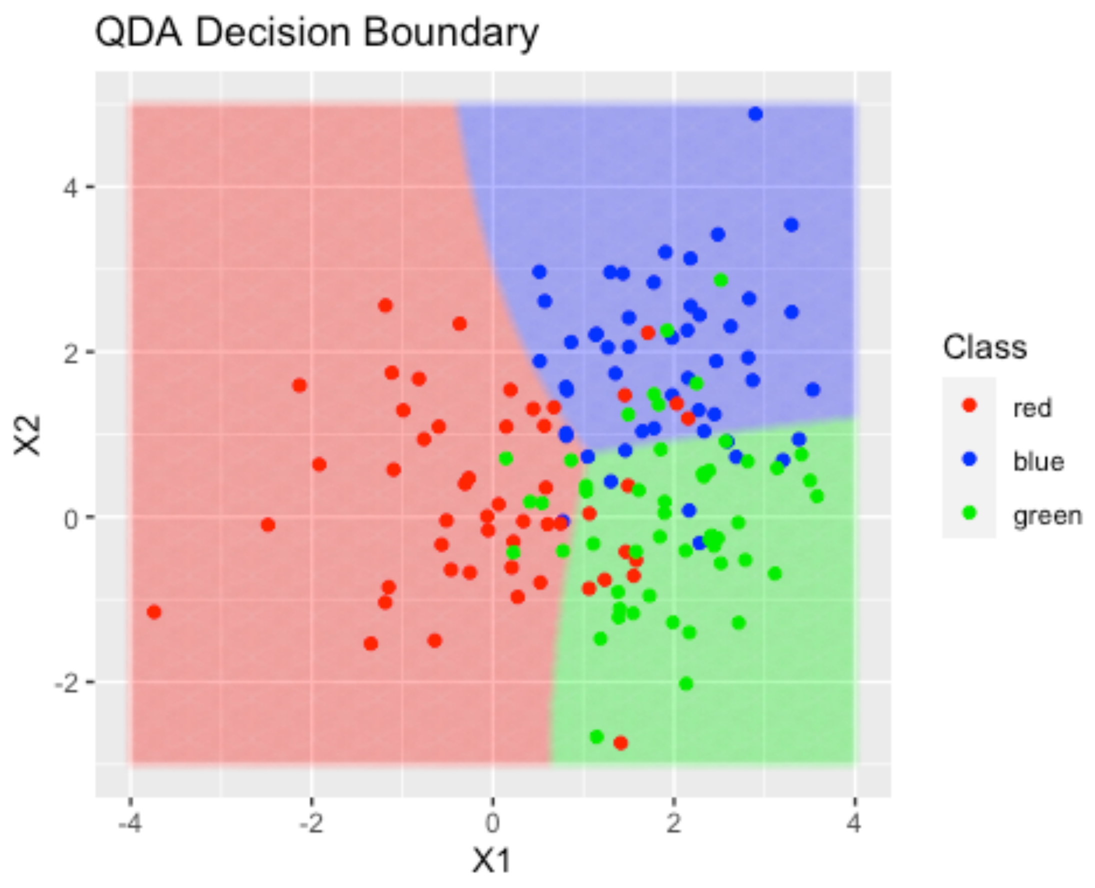
We see that for this particular choice ofμ_k and the random seed, Test error is less than training error. We see how LDA performs similar to Bayes classifier because the assumption of equal variance across all classes is satisfied. From above, we can see that the

LDA classifier is very close to the Bayes Classiefier, which the optimal case. Hence, LDA does a great job in classification for this example.

8. In this problem, you will apply quadratic discriminant analysis to the data from Q7.

a. Fit a quadratic discriminant analysis model to the training data from Q7, and make a plot that displays the observations as well as the QDA decision boundary (or boundaries) corresponding to this fitted model. Be sure to label which region(s) of the plot correspond to each class. How does the QDA decision boundary compare to the Bayes decision boundary that you computed in Q7(b)?

```
library(MASS)
qda.fit <- qda(Group~X1+X2, data=trainData7a)

test_data <- cbind(test_data, pred_qda = predict(qda.fit, test_data)$class)

ggplot() +
geom_point(data=test_data, aes(x=X1, y=X2, color=pred_qda), alpha=0.01)+
geom_point(data=trainData7a, aes(x=X1, y=X2, color=Group)) +
  labs(x = "X1",
       y="X2",
       title = "QDA Decision Boundary",
       color="Class")+
  scale_color_manual(values=colors)
```



b. Report the K × K confusion matrix for the QDA model on the training data, as well as the training error.

```
trainData7a <- cbind(trainData7a, pred_qda = predict(qda.fit, trainData7a)$cl
ass)

expected_value8b <- factor(trainData7a$pred_qda)
predicted_value8b <- factor(trainData7a$Group)
cm8b <- confusionMatrix(data=predicted_value8b, reference = expected_value8b)

cm8b

## Confusion Matrix and Statistics
##
##             Reference
## Prediction blue green red
##      blue    38     8   4
##      green    6    38   6
##      red      4     8  38
##
## Overall Statistics
##
##                Accuracy : 0.76
##                  95% CI : (0.6835, 0.8259)
##     No Information Rate : 0.36
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.64
##
##  Mcnemar's Test P-Value : 0.9029
##
## Statistics by Class:
##
##                      Class: blue Class: green Class: red
## Sensitivity               0.7917       0.7037     0.7917
## Specificity               0.8824       0.8750     0.8824
## Pos Pred Value            0.7600       0.7600     0.7600
## Neg Pred Value            0.9000       0.8400     0.9000
## Prevalence                0.3200       0.3600     0.3200
## Detection Rate            0.2533       0.2533     0.2533
## Detection Prevalence      0.3333       0.3333     0.3333
## Balanced Accuracy         0.8370       0.7894     0.8370
```

Training error = 0.24

c. Repeat (b), but this time using the test data generated in Q7. (That is, apply the model fit to the training data in (a) in order to calculate the test error.)

```
test_data_7e <- cbind(test_data_7e, pred_qda = predict(qda.fit, test_data_7e)
$class)

expected_value8c <- factor(test_data_7e$pred_qda)
```

```
predicted_value8c <- factor(test_data_7e$Group)
cm8c <- confusionMatrix(data=predicted_value8c, reference = expected_value8c)

cm8c

## Confusion Matrix and Statistics
##
##           Reference
## Prediction blue green red
##      blue    42     3   5
##      green    7    37   6
##      red      1     6  43
##
## Overall Statistics
##
##                Accuracy : 0.8133
##                  95% CI : (0.7416, 0.8722)
##     No Information Rate : 0.36
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.72
##
##  Mcnemar's Test P-Value : 0.2341
##
## Statistics by Class:
##
##                      Class: blue Class: green Class: red
## Sensitivity               0.8400       0.8043     0.7963
## Specificity               0.9200       0.8750     0.9271
## Pos Pred Value            0.8400       0.7400     0.8600
## Neg Pred Value            0.9200       0.9100     0.8900
## Prevalence                0.3333       0.3067     0.3600
## Detection Rate            0.2800       0.2467     0.2867
## Detection Prevalence      0.3333       0.3333     0.3333
## Balanced Accuracy         0.8800       0.8397     0.8617
```

Test error = 0.187

d. Compare your results in (b) and (c), and comment on your findings.

Similar to the 7th question, we see that the test error is less than the training error. We see that for this particular choice ofμ_k and the random seed, Test error is less than training error.

e. Which method had smaller training error in this example: LDA or QDA? Comment on your findings.

For this particular example, i.e, $\mu_1, \mu_2 and \mu_3$ and σ, we arrive at the same training error rates for both LDA and QDA. Comparing the decision boundaries of both the models, we see that both LDA and QDA give very similar boundaries to the Bayes decision boundary. We also observe that for this particular case, the QDA gives almost linear decision boundary. As we saw in Bayes Classifier, the optimal decision boundary is linear, so the QDA decision boundary not being very linear is a good thing for this example because we understand that it is very close to the optimal classification.

f. Which method had smaller test error in this example: LDA or QDA? Comment on your findings.

For this particular example, i.e, $\mu_1, \mu_2 and \mu_3$ and σ, we arrive at the same test error rates for both LDA and QDA. Comparing the decision boundaries of both the models, we see that both LDA and QDA give very similar boundaries to the Bayes decision boundary. We also observe that for this particular case, the QDA gives almost linear decision boundary. Even though the error is same, we still see a difference between misclassification by QDA and LDA. We also observe that for this particular case, the QDA gives almost linear decision boundary. As we saw in Bayes Classifier, the optimal decision boundary is linear, so the QDA decision boundary not being very linear is a good thing for this example because we understand that it is very close to the optimal classification.

Even though both LDA and QDA were close to Bayes Classifier, LDA was closer to Bayes than QDA was. However, while testing the performance, both the models perform equally good.

9. We have seen in class that the least squares regression estimator involves finding the coefficients $\beta_0, \beta_1, \ldots, \beta_p$ that minimize the quantity $\sum_{i=1}^{n}\left(y_i - \left(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}\right)\right)^2$. By contrast, the ridge regression estimator (which we will discuss in Chapter 6) involves finding the coefficients that minimize $\sum_{i=1}^{n}\left(y_i - \left(\beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}\right)\right)^2 + \lambda\left(\beta_0^2, \beta_1^2, \ldots, \beta_p^2\right)$ for some positive constant λ For simplicity, assume that $\beta_0 = 0$. Derive an expression for the ridge regression estimator, i.e. for the coefficient estimates $\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_p$

$$\hat{\beta}_{ridge} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - x_i^T \beta)^2 + \lambda \parallel \beta \parallel_2^2$$

$$\hat{\beta}_{ridge} = \underset{\beta}{\operatorname{argmin}} \parallel X\beta - Y \parallel_2^2 + \lambda \parallel \beta \parallel^2$$

$$\frac{\partial(\parallel X\beta - Y \parallel_2^2 + \lambda \parallel \beta \parallel^2)}{\partial \beta} = 0$$

$$\Rightarrow 2X^T(X\beta - Y) + 2\lambda\beta = 0$$

Rearranging,

$$(X^T X + \lambda I)\beta - X^T Y = 0$$

$$\hat{\beta}_{ridge} = (X^T X + \lambda I)^{-1} X^T Y$$