HW2

Peiran Chen

4/13/2022

1.

```
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
```

```
lm_1 <- lm(mpg ~ cylinders + displacement + horsepower + weight + acceleration + year + origin, data = A
kable(data.frame(Coefficients = lm_1$coefficients))
```

	Coefficients
(Intercept)	-17.2184346
cylinders	-0.4933763
displacement	0.0198956
horsepower	-0.0169511
weight	-0.0064740
acceleration	0.0805758
year	0.7507727
origin	1.4261405

Yes, there is a relationship between the predictors and the response by testing the null hypothesis of whether all the regression coefficients are zero. The F-statistic is far from 1 (with a small p-value), indicating evidence against the null hypothesis.

- When cylinder increases by one unit, mpg decreases by 0.493376.

- When displacement increases by one unit, mpg increases by 0.019896.
- When horsepower increases by one unit, mpg decreases by 0.016951.
- When weight increases by one unit, mpg decreases by 0.006474.
- When acceleration increases by one unit, mpg increases by 0.080576.
- When year increases by one unit, mpg increases by 0.750773.
- When origin increases by one unit, mpg increases by 1.426141.
- The intercept term is the mpg when all other coefficients are 0, -17.2184346.

b.

```
train_MSE <- mean(lm_1$residuals^2)
```

The train MSE in this linear model is 10.8474809.

c.

Since it's not hard to see that Origin = 3 means a Japanese car

```
prediction_1 <- predict(lm_1, data.frame(cylinders = 3, displacement = 100, horsepower = 85, weight = 3
```

The mileage my model predict for the given car is 28.3797758.

d.

```
mean(Auto$mpg[Auto$origin == 3]) - mean(Auto$mpg[Auto$origin == 1])
```

```
## [1] 10.41716
```

On average, holding all other covariates fixed, the difference between the **mpg** of a Japanese car and the **mpg** of an American car is -10.41716. That is, A Japanese car have 10.41716 higher mpg than an American car on average.

e.

```
10*lm_1$coefficients[4]
```

```
## horsepower
## -0.1695114
```

Hence, with 10 units increase in horsepower, we will see a 0.1695114 decrease in mpg.

2.

a.

Suppose we have y_i as the mpg of i^{th} vehicle. And

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \varepsilon_i = \begin{cases} \beta_0 + \beta_1 + \varepsilon_i \text{ if } i\text{th car is American made} \\ \beta_0 + \beta_2 + \varepsilon_i \text{ if } i\text{th car is European made} \\ \beta_0 + \varepsilon_i \text{ if } i\text{th car is Japanese made} \end{cases}$$

```
Auto_new <- Auto %>%
  mutate("American" = 1, "European" = 1)


Auto_new$American <- ifelse(Auto$origin == 1, 1, 0)
Auto_new$European <- ifelse(Auto$origin == 2, 1, 0)

lm(mpg~ American + European, data = Auto_new)
```

```
##
## Call:
## lm(formula = mpg ~ American + European, data = Auto_new)
##
## Coefficients:
## (Intercept)     American     European
##      30.451      -10.417       -2.848
```

Hence, the mpg prediction for a Japanese Car is 30.451, for American Car is 20.034, and 27.603 for European Car.

b.

Suppose we have y_i as the mpg of i^{th} vehicle. And

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \varepsilon_i = \begin{cases} \beta_0 + \beta_1 + \varepsilon_i \text{ if } i\text{th car is Japanese made} \\ \beta_0 + \beta_2 + \varepsilon_i \text{ if } i\text{th car is European made} \\ \beta_0 + \varepsilon_i \text{ if } i\text{th car is American made} \end{cases}$$

```
Auto_new <- Auto %>%
  mutate("Japanese" = 1, "European" = 1)


Auto_new$Japanese <- ifelse(Auto$origin == 3, 1, 0)
Auto_new$European <- ifelse(Auto$origin == 2, 1, 0)

lm(mpg~ Japanese + European, data = Auto_new)
```

```
##
## Call:
## lm(formula = mpg ~ Japanese + European, data = Auto_new)
##
```

```
## Coefficients:
## (Intercept)     Japanese     European
##      20.033       10.417        7.569
```

Hence, the mpg prediction for a Japanese Car is 30.45, for American Car is 20.033, and 27.602 for European Car.

c.

Suppose we have y_i as the mpg of i^{th} vehicle. And

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \varepsilon_i = \begin{cases} \beta_0 + \beta_1 - \beta_2 + \varepsilon_i \text{ if } i\text{th car is American made} \\ \beta_0 - \beta_1 + \beta_2 + \varepsilon_i \text{ if } i\text{th car is European made} \\ \beta_0 - \beta_1 - \beta_2 + \varepsilon_i \text{ if } i\text{th car is Japanese made} \end{cases}$$

```
Auto_new <- Auto %>%
  mutate("American" = 1, "European" = 1)


Auto_new$American <- ifelse(Auto$origin == 1, 1, -1)
Auto_new$European <- ifelse(Auto$origin == 2, 1, -1)

lm(mpg~ American + European, data = Auto_new)
```

```
##
## Call:
## lm(formula = mpg ~ American + European, data = Auto_new)
##
## Coefficients:
## (Intercept)     American     European
##      23.818       -5.209       -1.424
```

Hence, the mpg prediction for a Japanese Car is 30.451, for American Car is 20.033, and 27.603 for European Car.

d.

Suppose we have y_i as the mpg of i^{th} vehicle. And

$$y_i = \beta_0 + \beta_1 x_{i1} + \varepsilon_i = \begin{cases} \beta_0 + \beta_1 + \varepsilon_i \text{ if } i\text{th car is American made} \\ \beta_0 + 2\beta_1 + \varepsilon_i \text{ if } i\text{th car is European made} \\ \beta_0 + \varepsilon_i \text{ if } i\text{th car is Japanese made} \end{cases}$$

```
Auto_new["origin"][Auto_new["origin"] == 3] <- 0

lm(mpg~ origin, data = Auto_new)
```

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

```
##
## Coefficients:
## (Intercept)       origin
##      25.239       -1.845
```

Hence, the mpg prediction for a Japanese Car is 25.239, for American Car is 23.394, and 21.549 for European Car.

e.

My results from part a-c are consistent. However, for the last one, the result is different from previous ones. The reason for that is because the only one coefficient estimated would reflect a constrained effect where the expected mpg is incremented as a multiple of the dummy's regression coefficient.

3.

```
lm_3 <- lm(mpg~ origin + horsepower + origin*horsepower, data = Auto)
summary(lm_3)
```

```
##
## Call:
## lm(formula = mpg ~ origin + horsepower + origin * horsepower,
##     data = Auto)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.8206  -3.1504  -0.5536   2.3682  15.2386
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       26.79098    1.69728  15.785  < 2e-16 ***
## origin             7.87119    1.13907   6.910 2.00e-11 ***
## horsepower        -0.05942    0.01662  -3.574 0.000396 ***
## origin:horsepower -0.06338    0.01312  -4.832 1.95e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.424 on 388 degrees of freedom
## Multiple R-squared:  0.6812, Adjusted R-squared:  0.6788
## F-statistic: 276.4 on 3 and 388 DF,  p-value: < 2.2e-16
```

$$
\begin{aligned}
\text{mpg}_i &\approx \beta_0 + \beta_1 \times \text{origin}_i + \beta_2 \times \text{horsepower}_i + \beta_3 \times (\text{origin}_i \times \text{horsepower}_i) \\
&= \beta_0 + \beta_2 \times \text{horsepower}_i + (\beta_1 + \beta_3 \times \text{horsepower}_i) \times \text{origin}_i \\
&= \beta_0 + \beta_2 \times \text{horsepower}_i + \begin{cases} \beta_1 + \beta_3 \times \text{horsepower}_i \text{ if } i\text{th car is American made} \\ 2(\beta_1 + \beta_3 \times \text{horsepower}_i) \text{ if } i\text{th car is European made} \\ 3(\beta_1 + \beta_3 \times \text{horsepower}_i) \text{ if } i\text{th car is Japanese made} \end{cases} \\
&= \begin{cases} \beta_0 + \beta_1 + (\beta_2 + \beta_3) \times \text{horsepower}_i \text{ if } i\text{th car is American made} \\ \beta_0 + 2\beta_1 + (\beta_2 + 2\beta_3) \times \text{horsepower}_i) \text{ if } i\text{th car is European made} \\ \beta_0 + 3\beta_1 + (\beta_2 + 3\beta_3) \times \text{horsepower}_i) \text{ if } i\text{th car is Japanese made} \end{cases}
\end{aligned}
$$

Hence, we see that, with one unit increase in horsepower, the mpg for - Japanese car changes by -0.2495693. - American car changes by -0.1227999. - European car changes by -0.1861846.

4.

a.

Since we have the model, we can just plug in and get

$$\begin{aligned} Y &= \beta_0 + \beta_1 X_1 + \varepsilon \\ \hat{Y} &= \hat{\beta}_0 + \hat{\beta}_1 X_1 \\ \hat{Y} &= -165.1 + 4.8 X_1 \\ \text{Given that } X_1 &= 64, \\ \hat{Y} &= 142.1 \end{aligned}$$

Hence, the weight I predict for an individual who is 64 inches tall is 142.1.

b.

This time, we measure height in feet. That is, $X_1 = 12X_2$. And our model becomes

$$\begin{aligned} \hat{Y} &= \hat{\beta}_0 + \hat{\beta}_1 X_1 \\ \hat{Y} &= -165.1 + 4.8 \cdot 12 X_2 \\ \hat{Y} &= -165.1 + 57.6 X_2 \end{aligned}$$

Hence, we can see that $\beta_0^* = -165.1$, $\beta_1^* = 57.6$. And weight I predict for 5.333 feet tall is $\hat{Y} = -165.1 + 57.6 \cdot 5.333 = 142.0808$.

c.

$$\begin{aligned} Y &= \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \varepsilon \\ Y &= \hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2 + \hat{e} \end{aligned}$$

We want to minimize RSS s.t.

$$RSS = 0$$

$$\sum_{i=1}^{n} e_i^2 = 0$$

$$\sum_{i=1}^{n} (Y - \hat{\beta}_0 - \hat{\beta}_1 X_{1,i} - \hat{\beta}_2 X_{2,i})^2 = 0$$

$$\text{But we know that } X_2 = \frac{X_1}{12},$$

$$\sum_{i=1}^{n} (Y - \hat{\beta}_0 - \hat{\beta}_1 X_{1,i} - \hat{\beta}_2 \frac{X_{1,i}}{12})^2 = 0$$

$$\sum_{i=1}^{n} (Y - \hat{\beta}_0 - (\hat{\beta}_1 + \frac{\hat{\beta}_2}{12}) X_{1,i})^2 = 0$$

$$\text{If we set} \hat{\beta}_3 = \hat{\beta}_1 + \frac{\hat{\beta}_2}{12}$$

$$\sum_{i=1}^{n} (Y - \hat{\beta}_0 - \hat{\beta}_3 X_{1,i})^2 = 0$$

And the first order conditions are

$$\frac{\partial}{\partial \hat{\beta}_0} \sum_{i=1}^{n} (Y - \hat{\beta}_0 - \hat{\beta}_3 X_{1,i})^2 = 0$$
$$-2 \sum (Y - \hat{\beta}_0 - \hat{\beta}_3 X_{1,i}) = 0$$
$$\sum Y_i - n\hat{\beta}_0 - \hat{\beta}_3 \sum X_{1,i} = 0$$
$$\sum Y_i = n\hat{\beta}_0 + \hat{\beta}_3 \sum X_{1,i}$$
$$\bar{Y} = \hat{\beta}_0 + \hat{\beta}_3 \bar{X}_1$$
$$Y_i - \bar{Y} = (X_{1,i} - \bar{X})\hat{\beta}_3 + \hat{e}_i$$

Let $y_i = Y_i - \bar{Y}, x_i = X_i - \bar{X}$

$$y_i = \hat{\beta}_3 x_i + \hat{e}_i$$

Now, we can apply Least Squares

$$\frac{\partial}{\partial \hat{\beta}_3} \sum_{i=1}^{n} (y_i - \hat{\beta}_3 x_i)^2 = 0$$
$$2 \sum (y_i - \hat{\beta}_3 x_i)(-x_i) = 0$$
$$\sum (x_i y_i - \hat{\beta}_3 x_i^2) = 0$$
$$\sum x_i y_i - \hat{\beta}_3 \sum x_i^2 = 0$$
$$\hat{\beta}_3 = \frac{\sum x_i y_i}{\sum x_i^2}$$

Once $\hat{\beta}_3$ is estimated, we can estimate $\hat{\beta}_0$

$$\hat{\beta}_0 = \bar{Y} - \bar{X}_1 \hat{\beta}_3$$

Since there's collinearity between

$$X_1, X_2,$$

We can not seperate

$$\hat{\beta}_1 \text{ and } \hat{\beta}_2$$

And can only rely on that

$$\hat{\beta}_1 + \frac{\hat{\beta}_2}{12} = \hat{\beta}_3$$

d.

Since we know that $X1 = 12X_2$ from part b). The Structural multicollinearity existed betweenX_1, X_2 will reduce the precision of the estimated coefficients, And testing MSE will be greatest in this case. And remains the same for the rest two since units of measurement does not change the goodness of fit of our model. However, for **training MSE**, they will likely to be the same since we can effectively change our part c) regression into $Y + \hat{\beta}_0 + \hat{\beta}_3 X_{1,i} + e_i = 0$, where $\hat{\beta}_3 = \hat{\beta}_1 + \frac{\hat{\beta}_2}{12}$. Thus, it will give us the same result as the other two, hence they share the same **training MSE**.

5.

$$
\begin{aligned}
P(Y=1|X=x) &= P(Y=2|X=x) \\
\frac{\pi_1 f_1(x)}{\pi_1 f_1(x)+\pi_2 f_2(x)} &= \frac{\pi_2 f_2(x)}{\pi_1 f_1(x)+\pi_2 f_2(x)} \\
\pi_1 f_1(x) &= \pi_2 f_2(x) \\
\pi_1 \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} &= \frac{1}{4}\pi_2
\end{aligned}
$$

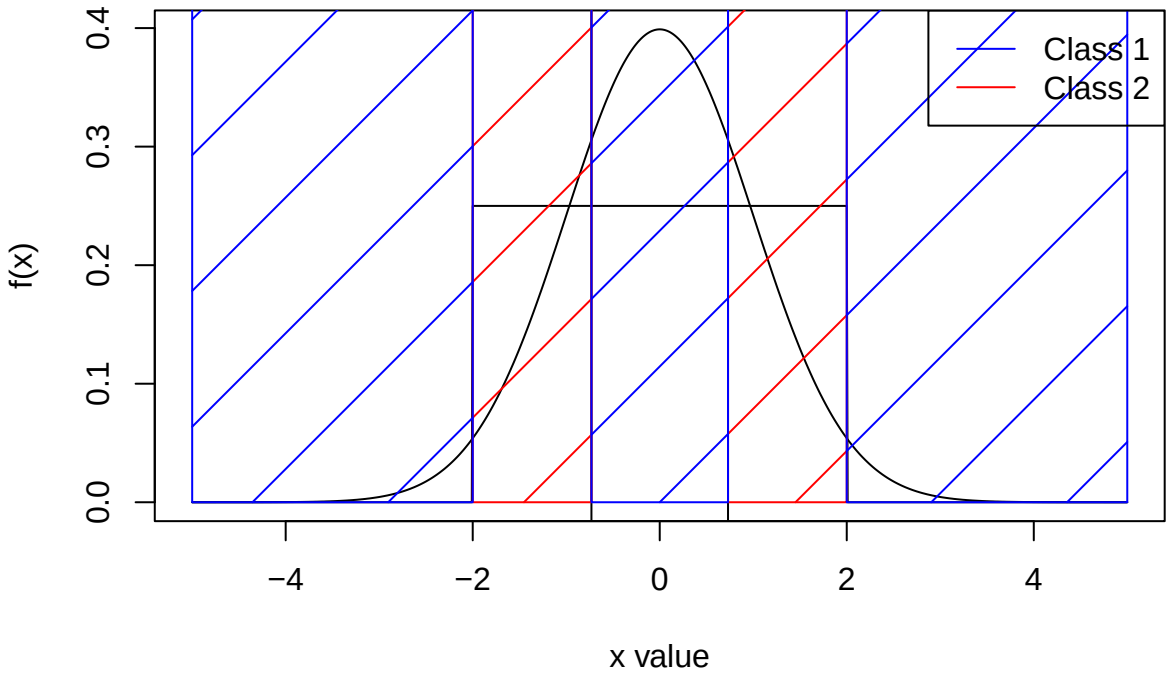
b.

Since we are now given the values for parameters, we can plug it in our decision boundary formula and see that, when $P(Y=1|X=x) \geqslant P(Y=2|X=x)$, we would classify it as Class one, and vice versa for class two.

$$
\begin{aligned}
P(Y=1|X=x) &\geqslant P(Y=2|X=x) \\
0.45 \cdot \frac{1}{2.506628} e^{-\frac{x^2}{2}} &\geqslant \frac{1}{4} \cdot 0.55 \\
|x| &\leqslant 0.7303212
\end{aligned}
$$

Hence, when x is in between ± 0.7303212. We would classify it as Class 1, and classify it to Class 2 elsewhere. However, if we look the other way around, we can see that if $|x| > 2$, $P(Y=2\Big||x|>2) = 0$. Hence, we would add the critera so that if $|x| > 2$ or $|x| \leqslant 0.7303212$, we would classify it as Class 1, other wise, we would classify it as Class 2.

```
x_base <- seq(-5, 5, by = 0.01)
plot(x_base, dnorm(x_base,0,1), type = "l",
     ylab = "f(x)",
     xlab = "x value")
lines(x_base, dunif(x_base, min = -2, max = 2))
abline(v = 0.7303212)
abline(v = -0.7303212)
rect(-2,0,-0.7303212,0.5,density = 2, col = "red")
rect(0.7303212,0,2,0.5,density = 2, col = "red")
rect(-0.7303212,0,0.7303212,0.5,density = 2, col = "blue")
rect(-5,0,-2,0.5,density = 2, col = "blue")
rect(2,0,5,0.5,density = 2, col = "blue")

legend("topright",
       c("Class 1","Class 2"),
       col = c("blue","red"),
       lty = 1:1)
```



###

c.

We can estimate these by

$$\hat{\mu}_1 = \frac{1}{n_1} \sum_{i:y_i=1} x_i$$
$$\hat{\pi}_1 = \frac{n_1}{n_1 + n_2}$$
$$\hat{\sigma}_1 = \frac{1}{n-1} \sum_{k=1}^{2} \sum_{i:y_i=k} (x_i - \hat{\mu}_1)^2$$

d.

$$P(Y = 1|X = x_0) = \frac{\hat{\pi}_1 \hat{f}_1(x_0)}{\hat{\pi}_1 \hat{f}_1(x_0) + \hat{\pi}_2 \hat{f}_2(x_0)}$$
$$= \frac{\hat{\pi}_1 \frac{1}{\hat{\sigma}\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x_0 - \hat{\mu}}{\hat{\sigma}})^2}}{\hat{\pi}_1 \frac{1}{\hat{\sigma}\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x_0 - \hat{\mu}}{\hat{\sigma}})^2} + \frac{1}{4}\hat{\pi}_2}$$

6.

a.

$$\log\left[\frac{p(x)}{1-p(x)}\right] = 0.7$$
$$\beta_0 + \beta_1 x_1 + ... + \beta_p x_p = 0.7$$

Hence,

$$\begin{aligned} P(Y=1|X=x) &= \frac{\exp(\beta_0 + \beta_1 x_1 + ... + \beta_p x_p)}{1+\exp(\beta_0 + \beta_1 x_1 + ... + \beta_p x_p)} \\ &= \frac{\exp(0.7)}{1+\exp(0.7)} \\ &= 0.6682 \end{aligned}$$

b.

$$P(Y=1|X=x^*) = \frac{\exp\left(\beta_0 + \beta_1 x_1^* + \beta_2 x_2^* + ... + \beta_p x_p^*\right)}{1+\exp\left(\beta_0 + \beta_1 x_1^* + \beta_2 x_2^* + ... + \beta_p x_p^*\right)}$$

and it's odds equals

$$\begin{aligned} odds &= \beta_0 + \beta_1(x_1+1) + \beta_2(x_2-1) + \beta_3(x_3+2) + ... + \beta_p x_p^* \\ &= \beta_0 + \beta_1 x_1 + \beta_1 + \beta_2 x_2 - \beta_2 + \beta_3 x_3 + 2\beta_3 + ... + \beta_p x_p^* \\ &= 0.7 + \beta_1 - \beta_2 + 2\beta_3 \end{aligned}$$

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

7.

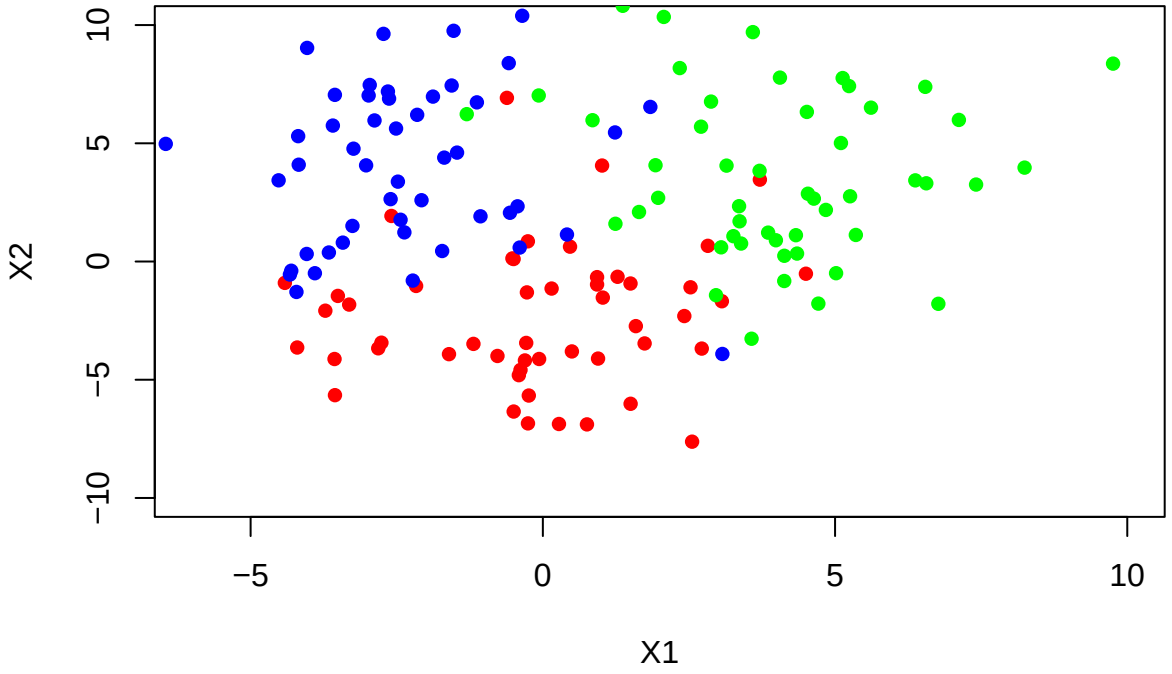
a.

```
n <- 50
rho <- 0.7

class_1 <- data.frame(X1 = rnorm(n, 0, 2), X2 = rnorm(n, -2, 3))
class_2 <- data.frame(X1 = rnorm(n, 4, 2), X2 = rnorm(n, 4, 3))
class_3 <- data.frame(X1 = rnorm(n, -2, 2), X2 = rnorm(n, 5, 3))

# Convert data.frame into matrix
C1 <- data.matrix(class_1)
C2 <- data.matrix(class_2)
C2 <- data.matrix(class_3)

mu1 <- matrix(c(0, -2), 2, 1)
mu2 <- matrix(c(4, 4), 2, 1)
mu3 <- matrix(c(-2, 5), 2, 1)

sigma <- matrix(c(4, 0, 0, 9), 2, 2)
```

My choice for $\mu_1 = \begin{bmatrix} 0 \\ -2 \end{bmatrix}; \mu_2 = \begin{bmatrix} 4 \\ 4 \end{bmatrix}; \mu_3 = \begin{bmatrix} -2 \\ 5 \end{bmatrix}$. And $\sum = \begin{bmatrix} 4 & 0 \\ 0 & 9 \end{bmatrix}$.

b.

```
plot(class_1$X1, class_1$X2,
     col = "red",
     pch = 16,
     xlim = c(-6, 10),
     ylim = c(-10,10),
     xlab = "X1",
     ylab = "X2")
points(class_2$X1, class_2$X2, col = "green", pch = 16)
points(class_3$X1, class_3$X2, col = "blue", pch = 16)
```




```
# Calculate Bayes Devision Boundary
```

And for the Bayes decision boundary, we have to calculate the below function:

$$
\begin{aligned}
X^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k &= X^T\Sigma^{-1}\mu_l - \frac{1}{2}\mu_l^T\Sigma^{-1}\mu_l \quad \forall k \neq l \\
X^T(\Sigma^{-1}\mu_k - X^T\Sigma^{-1}\mu_l) &= \frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_l^T\Sigma^{-1}\mu_l \\
X^T(\Sigma^{-1}\mu_k - \Sigma^{-1}\mu_l) &= \frac{1}{2}(\mu_k^T\Sigma^{-1}\mu_k - \mu_l^T\Sigma^{-1}\mu_l) \\
X^T &= \frac{1}{2}(\mu_k^T\Sigma^{-1}\mu_k - \mu_l^T\Sigma^{-1}\mu_l)(\Sigma^{-1}\mu_k - \Sigma^{-1}\mu_l)^{-1} \\
X &= \frac{1}{2}\left((\mu_k^T\Sigma^{-1}\mu_k - \mu_l^T\Sigma^{-1}\mu_l)(\Sigma^{-1}\mu_k - \Sigma^{-1}\mu_l)^{-1}\right)^T
\end{aligned}
$$

However, since that $\Sigma^{-1}\mu_k - \Sigma^{-1}\mu_l$ is a 2×1 matrix, it can not be inverted. And we would use

$$
X^T(\Sigma^{-1}\mu_k - \Sigma^{-1}\mu_l) = \frac{1}{2}(\mu_k^T\Sigma^{-1}\mu_k - \mu_l^T\Sigma^{-1}\mu_l)
$$

to solve for X. And let

$$A = \Sigma^{-1}\mu_k - \Sigma^{-1}\mu_l$$
$$b = \frac{1}{2}(\mu_k^T\Sigma^{-1}\mu_k - \mu_l^T\Sigma^{-1}\mu_l)$$
$$\text{Thus, } X^T A = b$$

```
A <- sigma %*% mu1 - sigma %*% mu2
b <- 0.5 * (t(mu1) %*% solve(sigma) %*% mu1 - t(mu2) %*% solve(sigma) %*% mu2)
```

Hence, we have

$$\begin{bmatrix} X_1 & X_2 \end{bmatrix} \begin{bmatrix} -41.2 \\ -70.8 \end{bmatrix} = -1.568627$$

So, we have $-41.2X_1 - 70.8X_2 + 1.568627 = 0$. And we can repeat the process for other decisions.

```
A <- sigma %*% mu2 - sigma %*% mu3
b <- 0.5 * (t(mu2) %*% solve(sigma) %*% mu2 - t(mu3) %*% solve(sigma) %*% mu3)
slope <- -1 *A[1]/A[2]
intersect <- b/A[2]
slope
```

```
## [1] 2.666667
```

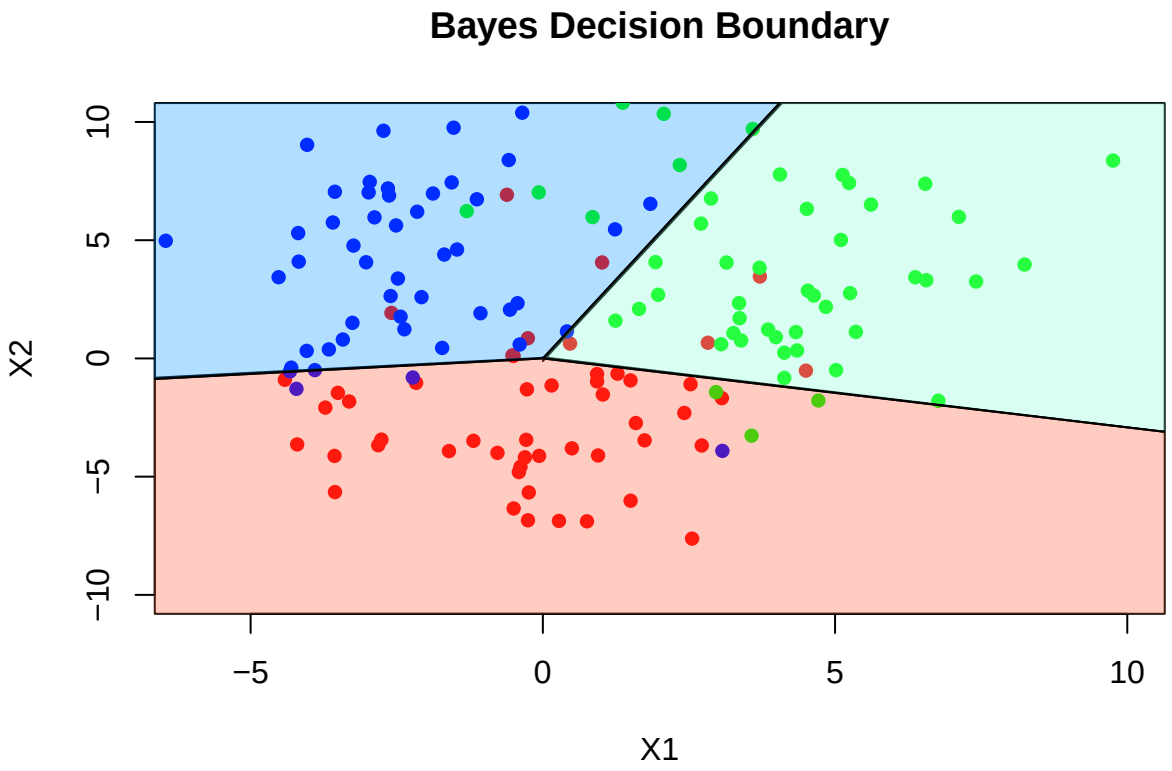
```
intersect
```

```
##             [,1]
## [1,] -0.1111111
```

And we can plot this as:

```
x_red_green <- seq(0, 100, by = 0.001)
x_red_blue <- seq(-10, 0.1, by = 0.01)
x_green_blue <- seq(0, 100, by = 0.01)

y_red_green <- -0.2962963*x_red_green + 0.04938272
y_red_blue <- 0.1269841*x_red_blue + 0.02645503
y_green_blue <- 2.666667*x_green_blue - 0.1111111

plot(class_1$X1, class_1$X2,
     col = "red",
     pch = 16,
     xlim = c(-6, 10),
     ylim = c(-10,10),
     xlab = "X1",
     ylab = "X2",
     main = "Bayes Decision Boundary")
points(class_2$X1, class_2$X2, col = "green", pch = 16)
points(class_3$X1, class_3$X2, col = "blue", pch = 16)
lines(x_red_green, y_red_green) # Red Green
lines(x_red_blue, y_red_blue) # Red Blue
```

```
lines(x_green_blue, y_green_blue) # Blue Green

polygon(x = c(0, 4.5, 12, 12),
        y = c(0, 12, 12, -3.5),
        col = rgb(127/255, 1, 212/255, alpha = 0.3))
polygon(x = c(-8, -8, 4.5, 0),
        y = c(-1.05, 12, 12, 0),
        col = rgb(0, 150/255, 1, alpha = 0.3))
polygon(x = c(-8, -8, 0, 12, 12),
        y = c(-12, -1.05,0, -3.5, -12),
        col = rgb(1, 87/255, 51/255, alpha = 0.3))
```

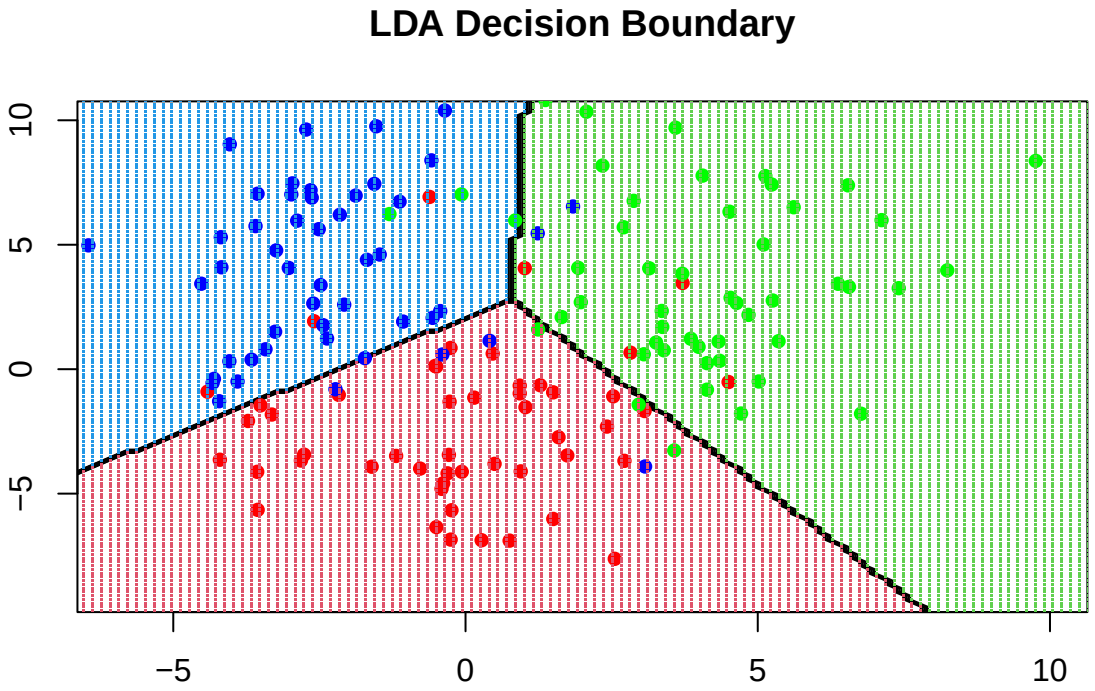


d.

To get the LDA Decision Boundary, we can try to

```
train <- tibble(
  X1 = c(class_1$X1, class_2$X1, class_3$X1),
  X2 = c(class_1$X2, class_2$X2, class_3$X2),
  class = c(rep("red", 50), rep("green", 50), rep("blue", 50))
)

train <- data.frame(train)
train_lda <- lda(class ~ ., data = train)
grid <- seq(-10, 20,length = 200)
grid_2d <- expand.grid(X1=grid, X2=grid)
grid_pred_lda <- as.numeric(predict(train_lda, newdata=grid_2d)$class)
grid_pred_lda <- matrix(grid_pred_lda, ncol = 200)
```

```
contour(grid, grid, grid_pred_lda,
        xlim = c(-6, 10), ylim = c(-9, 10),
        drawlabels = F, lty = 1, col = "black",
        main = "LDA Decision Boundary")
points(train[, 1], train[, 2], col = train$class,
       pch = 16)
points(grid_2d, pch=".", cex=1.2, col=ifelse(grid_pred_lda == 3, 2,
                                             ifelse(grid_pred_lda == 2, 3, 4)))
```



When I compare the LDA Boundary to the Bayes one, I would say they are pretty similar.

d.

```
# Plot the Confusion Matrix for LDA
table(predict(train_lda,type="class")$class, train$class)
```

```
##
##         blue green red
##   blue    43     3   3
##   green    2    44   4
##   red      5     3  43
```

```
training_error_lda <- mean(predict(train_lda,type="class")$class != train$class)
```

The Training Error is 0.1333333.

e.

```
test_class_1 <- data.frame(X1 = rnorm(n, 0, 2), X2 = rnorm(n, -2, 3))
test_class_2 <- data.frame(X1 = rnorm(n, 4, 2), X2 = rnorm(n, 4, 3))
test_class_3 <- data.frame(X1 = rnorm(n, -2, 2), X2 = rnorm(n, 5, 3))

test <- tibble(
  X1 = c(test_class_1$X1, test_class_2$X1, test_class_3$X1),
  X2 = c(test_class_1$X2, test_class_2$X2, test_class_3$X2),
  class = c(rep("red", 50), rep("green", 50), rep("blue", 50))
)

test <- data.frame(test)
test_lda <- predict(train_lda, newdata = test)
table(test_lda$class, test$class)
```

```
##
##         blue green red
##   blue    44     3   8
##   green    4    42   3
##   red      2     5  39
```

```
testing_error_lda <- mean(test_lda$class != test$class)
```

And our Test Error is 0.1666667.

f.

And the difference between the training and testing error is caused by the irreducible error term, since just like other Statistical Learning Methods, LDA is trying to minimize the Reducible Error term. And the difference between these two error term would cancel out the reducible error term, since we have the same model, and the difference can only be caused by the irreducible error term, that is, the randomness in our dataset.

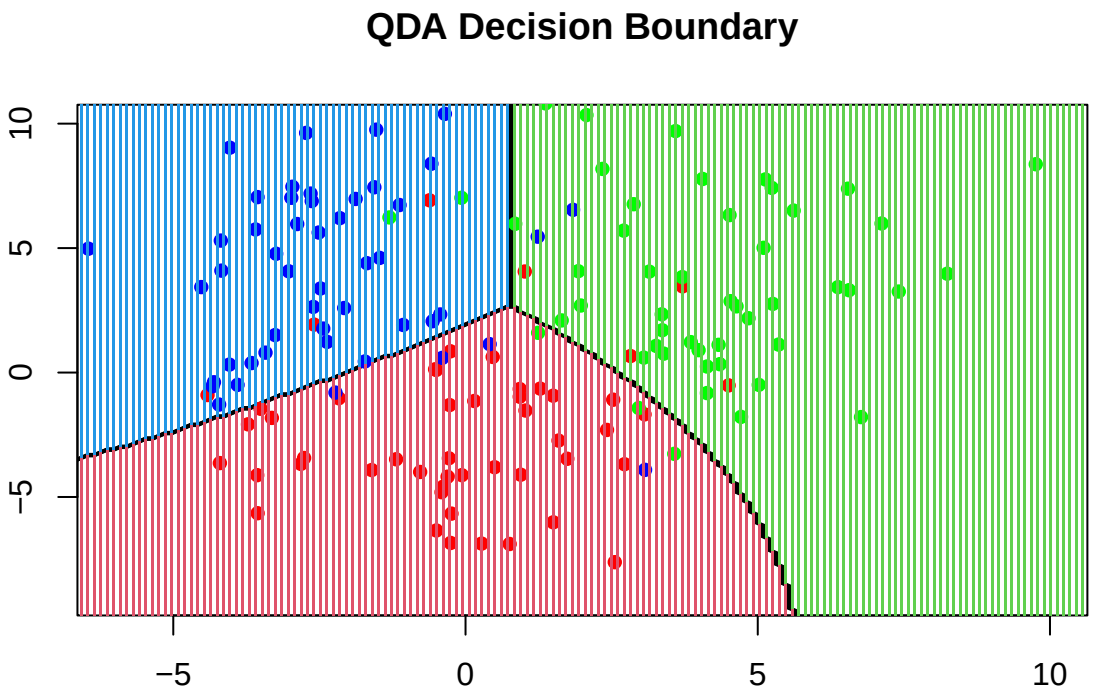
$$\text{Error} = \text{Reducible Error} + \text{Irreducible Error}$$

8.

a.

```
train_qda <- qda(class ~ ., data = train)
grid <- seq(-10, 12,length = 200)
grid_2d <- expand.grid(X1=grid, X2=grid)
grid_pred_qda <- as.numeric(predict(train_qda, newdata=grid_2d)$class)
grid_pred_qda <- matrix(grid_pred_qda, ncol = 200)
contour(grid, grid, grid_pred_qda,
        xlim = c(-6, 10), ylim = c(-9, 10),
        drawlabels = F, lty = 1, col = "black",
```

```
        main = "QDA Decision Boundary")
points(train[, 1], train[, 2], col = train$class,
       pch = 16)
points(grid_2d, pch=".", cex=1.2, col=ifelse(grid_pred_qda == 3, 2,
                                             ifelse(grid_pred_qda == 2, 3, 4)))
```



b.

```
table(predict(train_qda,type="class")$class, train$class)
```

```
##
##         blue green red
##   blue    44     2   3
##   green    2    45   4
##   red      4     3  43
```

```
training_error_qda <- mean(predict(train_qda,type="class")$class != train$class)
```

This time, the training error is 0.12.

c.

```
test_qda <- predict(train_qda, newdata = test)
table(test_qda$class, test$class)
```

```
##
##         blue green red
##   blue    44     3   9
##   green    4    42   4
##   red      2     5  37
```

```
testing_error_qda <- mean(test_qda$class != test$class)
```

This time, the testing error is 0.18.

d.

And the difference between the training and testing error is caused by the irreducible error term, since just like other Statistical Learning Methods, QDA is trying to minimize the Reducible Error term. And the difference between these two error term would cancel out the reducible error term, since we have the same model, and the difference can only be caused by the irreducible error term, that is, the randomness in our dataset.

$$\text{Error} = \text{Reducible Error} + \text{Irreducible Error}$$

e & f.

```
kable(tibble(
  " " = c("LDA", "QDA"),
  "Test Error" = c(testing_error_lda, testing_error_qda),
  "Train Error" = c(training_error_lda, training_error_qda),
))
```

	Test Error	Train Error
LDA	0.1666667	0.1333333
QDA	0.1800000	0.1200000

We can see from the above Table that, in my case, LDA has lower Test Error, and QDA has lower training error. The reason for that is with to do with flexibility of our methods of approach. Since the involvement of Quadratic terms, QDA would have more flexibility than LDA. Thus it might overfit the data and cause it to fit "too hard" to a point that it involves some of the irreducible error as part of the fitting. Hence it tends to have less training error. And that "overfitting" result in QDA have higher Testing Error than the LDA.

9. Extra Credits

We can use the linear algebra approach to solve this, that is, we can first convert the equation into the form of matrices.

Let $Y = [y_1\ y_2\ y_3\ ...y_n]^T$, $\beta = [\beta_0\ \beta_1\ \beta_2\ \beta_3\ ...\beta_n]^T$.$X = \begin{bmatrix} 1 & x_{11} & ... & x_{1p} \\ 1 & x_{21} & ... & x_{2p} \\ ... & ... & ... & ... \\ 1 & x_{n1} & ... & x_{np} \end{bmatrix}$ And $\lambda = I\lambda$.

$$\frac{\partial}{\partial\beta}\sum_{i=1}^{n}(y_i-(\beta_0+\beta_1x_{i1}+...+\beta_px_{ip}))^2+\lambda(\beta_1^2+...+\beta_p^2)=0$$

$$\begin{aligned}
\frac{\partial}{\partial\beta}(Y-X\beta)^T(Y-X\beta)+\lambda I\beta^T\beta&=0\\
-2X^T(Y-X\beta)+2\lambda I\beta&=0\\
X^T(Y-X\beta)-\lambda I\beta&=0\\
X^TY-X^TX\beta-\lambda I\beta&=0\\
\beta(X^TX+\lambda I)&=X^TY\\
\hat{\beta}&=(X^TX+\lambda I)^{-1}X^TY
\end{aligned}$$