DATA 558: Statistical Machine Learning
Homework 2

Abhishek Saini

April 2022

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

   (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.
   The qualitative variable 'origin' is encoded using one hot encoding the factors The variable factor(origin)2 is 1 when the car is of European origin and 0 otherwise. The variable factor(origin)3 is 1 when the car is of Japanese origin and 0 otherwise.

```
library(ggplot2)
# library(class)
library(ISLR2)

# 1 a) encode origin as a factor and fit the linear model
auto_df = Auto
auto_df <- subset(auto_df, select = -c(name))
auto_df$origin = factor(auto_df$origin)
lm1.fit = lm(mpg ~ ., data = auto_df)
lm1.summary = summary(lm1.fit)
lm1.summary

Call:
lm(formula = mpg ~ ., data = auto_df)

Residuals:
    Min      1Q  Median      3Q     Max
-9.0095 -2.0785 -0.0982  1.9856 13.3608

Coefficients:
```

```
                Estimate Std. Error t value Pr(>|t|)
(Intercept)  -1.795e+01  4.677e+00  -3.839 0.000145 ***
cylinders    -4.897e-01  3.212e-01  -1.524 0.128215
displacement  2.398e-02  7.653e-03   3.133 0.001863 **
horsepower   -1.818e-02  1.371e-02  -1.326 0.185488
weight       -6.710e-03  6.551e-04 -10.243  < 2e-16 ***
acceleration  7.910e-02  9.822e-02   0.805 0.421101
year          7.770e-01  5.178e-02  15.005  < 2e-16 ***
origin2       2.630e+00  5.664e-01   4.643 4.72e-06 ***
origin3       2.853e+00  5.527e-01   5.162 3.93e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.307 on 383 degrees of freedom
Multiple R-squared:  0.8242,        Adjusted R-squared:  0.8205
F-statistic: 224.5 on 8 and 383 DF,  p-value: < 2.2e-16
```

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

The training set mean squared error is given as:

$$MSE = RSS/(n-p-1) = 10.93314$$

```
sum(lm1.summary$residuals^2)/lm1.summary$df[2]
[1] 10.93314
```

(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?

The predicted mileage is 27.89483.

```
predict_df = data.frame(cylinders = c(3), displacement = c(100.0), horsepower = c(85)
predict_df$origin = factor(predict_df$origin)
predict(lm1.fit, predict_df)
27.89483
```

(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?

This is given by the coefficient of factor(origin)3 in our model which is equal to 2.853.

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

This would be 10 times the coefficient associated with horsepower $= 10\hat{\beta}_{horsepower} = 10 \times -0.01818 = -0.1818$

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?

$$y_i = \beta_J + \beta_A x_{iA} + \beta_E x_{iE} + \epsilon_i = \begin{cases} \beta_J + \beta_A + \epsilon_i & \text{if car is American} \\ \beta_J + \beta_E + \epsilon_i & \text{if car is European} \\ \beta_J + \epsilon_i & \text{if car is Japanese} \end{cases}$$

The coefficient estimates are: $\hat{\beta}_J = 30.451, \hat{\beta}_A = -10.417, \hat{\beta}_E = -2.848$.
The predicted mpg are:
Japanese car: 30.45063
American car: 20.03347
European car: 27.60294

```
# 2 a)
auto_df = Auto
auto_df$origin = factor(auto_df$origin)
auto_df$origin = relevel(auto_df$origin, ref = 3)    # set Japanese as the baseline

lm2.fit = lm(mpg ~ origin, data = auto_df)
lm2.fit

predict_df = data.frame(origin = c(1, 2, 3))
predict_df$origin = factor(predict_df$origin)
predict(lm2.fit, predict_df)
```

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

Coefficients:
(Intercept)      origin1      origin2
     30.451      -10.417       -2.848

       1        2        3
20.03347 27.60294 30.45063
```

(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?

$$y_i = \beta_A + \beta_J x_{iJ} + \beta_E x_{iE} + \epsilon_i = \begin{cases} \beta_A + \beta_J + \epsilon_i & \text{if car is Japanese} \\ \beta_A + \beta_E + \epsilon_i & \text{if car is European} \\ \beta_A + \epsilon_i & \text{if car is American} \end{cases}$$

The coefficient estimates are: $\hat{\beta}_J = 10.417, \hat{\beta}_A = 20.033, \hat{\beta}_E = 7.569$.

The predicted mpg are:

Japanese car: 30.45063

American car: 20.03347

European car: 27.60294

```
# 2 b)
auto_df = Auto
auto_df$origin = factor(auto_df$origin)
auto_df$origin = relevel(auto_df$origin, ref = 1)    # set American as the baseline

lm2.fit = lm(mpg ~ origin, data = auto_df)
lm2.fit

predict_df = data.frame(origin = c(1, 2, 3))
predict_df$origin = factor(predict_df$origin)
predict(lm2.fit, predict_df)

Call:
lm(formula = mpg ~ origin, data = auto_df)

Coefficients:
(Intercept)      origin2      origin3
     20.033        7.569       10.417

       1          2          3
20.03347 27.60294 30.45063
```

(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?

$$y_i = \beta_A + \beta_J x_{iJ} + \beta_E x_{iE} + \epsilon_i = \begin{cases} \beta_A + \beta_J - \beta_E + \epsilon_i & \text{if car is Japanese} \\ \beta_A - \beta_J + \beta_E + \epsilon_i & \text{if car is European} \\ \beta_A - \beta_J - \beta_E + \epsilon_i & \text{if car is American} \end{cases}$$

The coefficient estimates are: $\hat{\beta}_J = 5.209, \hat{\beta}_A = 29.027, \hat{\beta}_E = 3.785$.

The predicted mpg are:

Japanese car: 30.45063

American car: 20.03347

European car: 27.60294

```
# 2 c)
auto_df = Auto
auto_df$japanese = ifelse(auto_df$origin==3, 1, -1)
auto_df$european = ifelse(auto_df$origin==2, 1, -1)

lm2.fit = lm(mpg ~ japanese + european, data = auto_df)
lm2.fit

predict_df = data.frame(origin = c(1, 2, 3))
predict_df$japanese = ifelse(predict_df$origin==3, 1, -1)
predict_df$european = ifelse(predict_df$origin==2, 1, -1)
predict(lm2.fit, predict_df)
```

```
Call:
lm(formula = mpg ~ japanese + european, data = auto_df)

Coefficients:
(Intercept)     japanese     european
     29.027        5.209        3.785

       1        2        3
20.03347 27.60294 30.45063
```

(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?

$$y_i = \beta_0 + \beta_1 x_{i\_origin} + \epsilon_i = \begin{cases} \beta_0 + 2\beta_1 + \epsilon_i & \text{if car is European} \\ \beta_0 + \beta_1 + \epsilon_i & \text{if car is American} \\ \beta_0 + \epsilon_i & \text{if car is Japanese} \end{cases}$$

The coefficient estimates are: $\hat{\beta}_0 = 25.239, \hat{\beta}_1 = -1.845$
The predicted mpg are:
Japanese car: 25.23947
American car: 23.39414
European car: 21.54880

```
# 2 d)
auto_df = Auto
auto_df$origin = auto_df$origin %% 3

lm2.fit = lm(mpg ~ origin, data = auto_df)
lm2.fit

predict_df = data.frame(origin = c(1, 2, 3))
predict_df$origin = predict_df$origin %% 3
predict(lm2.fit, predict_df)

Call:
lm(formula = mpg ~ origin, data = auto_df)

Coefficients:
(Intercept)       origin
     25.239       -1.845


       1        2        3
23.39414 21.54880 25.23947
```

(e) Comment on your results in (a)-(d).
The prediction results in (a) to (c) exactly match with each other. This is because the information that the model receives remains exactly the same. The only difference is that the model sees a transformed version of the data and thus the parameters learned are also transformed accordingly. The prediction remains the same as the predictions are invariant to linear transformations of the input.
In (d) the model we are learning is different as it has one fewer parameter so the predictions

are likely to differ. However, it is not a wise strategy to encode categorical variables as ordered numbers because this enforces an ordering on the features when there isn't any. This could cause the model to not fit the data correctly.

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?

$$
\begin{aligned}
mpg_i &\approx \beta_0 + \beta_1 \times is\_european + \beta_2 \times is\_japanese + \beta_3 \times hp + \beta_4 \times is\_european \times hp \\
&\qquad + \beta_5 \times is\_japanese \times hp \\
&= \begin{cases} (\beta_0 + \beta_1) + (\beta_3 + \beta_4) \times hp & \text{if car is European} \\ (\beta_0 + \beta_2) + (\beta_3 + \beta_5) \times hp & \text{if car is Japanese} \\ \beta_0 + \beta_3 \times hp & \text{if car is American} \end{cases}
\end{aligned}
$$

The coefficient estimates are: $\hat{\beta}_0 = 34.4765, \hat{\beta}_1 = 10.9972, \hat{\beta}_2 = 14.3397, \hat{\beta}_3 = -0.1213, \hat{\beta}_4 = -0.1005, \hat{\beta}_5 = -0.1087$.
On average, mpg of European car increases by $(\hat{\beta}_3 + \hat{\beta}_4) = -0.1213 - 0.1005 = -0.2218$ units for a one-unit increase in horsepower.
On average, mpg of Japanese car increases by $(\hat{\beta}_3 + \hat{\beta}_5) = -0.1213 - 0.1087 = -0.23$ units for a one-unit increase in horsepower.
On average, mpg of American car increases by $\hat{\beta}_3 = -0.1213$ units for a one-unit increase in horsepower.

```
# 3
auto_df = Auto
auto_df$origin = factor(auto_df$origin)

lm3.fit = lm(mpg ~ origin*horsepower, data = auto_df)
lm3.fit

Call:
lm(formula = mpg ~ origin * horsepower, data = auto_df)

Coefficients:
      (Intercept)            origin2            origin3          horsepower  origin2:ho
          34.4765            10.9972            14.3397             -0.1213
origin3:horsepower
          -0.1087
```

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

(a) Suppose that you measure height in inches (X_1), 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?
I would predict weight $Y = \hat{\beta}_0 + \hat{\beta}_1 \times X_1 = -165.1 + 4.8 \times 64 = 142.1$ units

(b) Now suppose that you want to measure height in feet (X_2) 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)?
Since the new feature X_2 is just a scaled transform of X_1, $X_2 = X_1/12$, β_1^* will be adjusted by the corresponding amount, $\beta_1^* = \beta_1 \times 12 = 4.8 \times 12 = 57.6$. β_0^* will be the same as β_0, $\beta_0^* = \beta_0 = -165.1$. Weight predicted will be the same as before $= 142.1$ units since the least squares predictions are invariant to linear transformations of the input data.

(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.
Since the features are not linearly independent there will not be a unique least squares solution. To find the general solution, let's rewrite the equation of the model.

$$\begin{aligned} Y &= \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon \\ &= \beta_0 + \beta_1 X_1 + \beta_2 X_1/12 + \epsilon \\ &= \beta_0 + \left(\beta_1 + \frac{\beta_2}{12}\right) X_1 + \epsilon \end{aligned}$$

The least squares solution to the above equation will be

$$\begin{aligned} \hat{\beta}_0 &= -165.1 \\ \hat{\beta}_1 + \frac{\hat{\beta}_2}{12} &= 4.8 \end{aligned}$$

(d) How do the (training set) mean squared errors compare for three models fit in (a)–(c)?
The means squared errors in (a)-(c) will be the same because the least squares predictions are invariant to linear transformations of the input data.

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 $N(\mu, \sigma^2)$ 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.

$$P(Y = 1|X = x) = \frac{\pi_1 f_1(X = x|Y = 1)}{\pi_1 f_1(X = x|Y = 1) + \pi_2 f_2(X = x|Y = 2)} \tag{1}$$
$$P(Y = 2|X = x) = \frac{\pi_1 f_2(X = x|Y = 2)}{\pi_1 f_1(X = x|Y = 1) + \pi_2 f_2(X = x|Y = 2)} \tag{2}$$

Equating (1) and (2), we get,

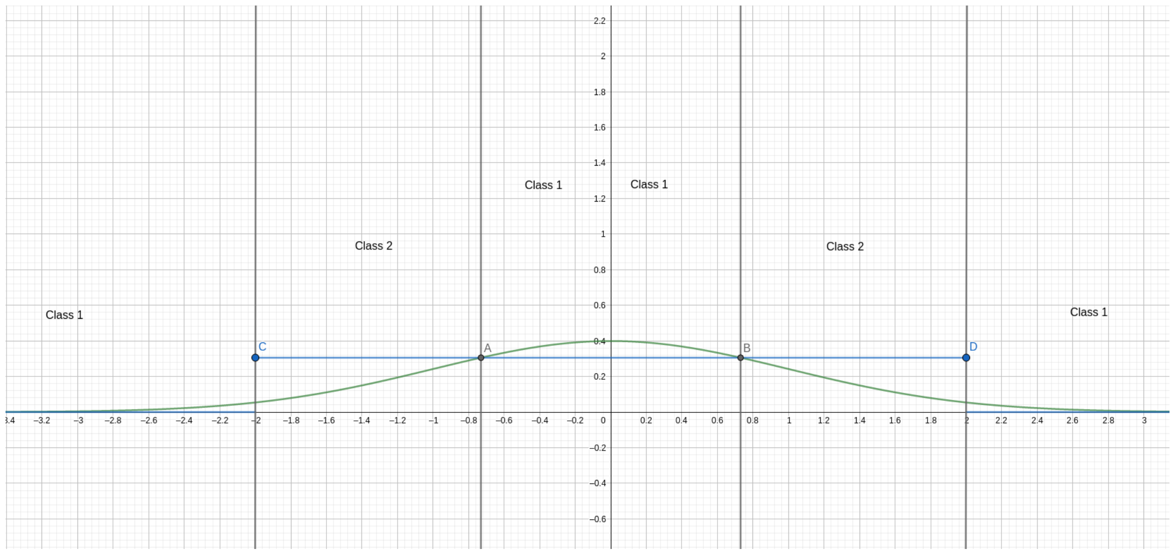
$$\begin{aligned}
\pi_1 f_1(X = x|Y = 1) &= \pi_2 f_2(X = x|Y = 2) \\
\pi_1 \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right) &= \pi_2 \frac{1}{4} \qquad\qquad x \in [-2, 2] \\
\exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right) &= \frac{\pi_2\sqrt{2\pi\sigma^2}}{\pi_1 4} \\
\frac{-(x-\mu)^2}{2\sigma^2} &= \log\left(\frac{\pi_2\sqrt{2\pi\sigma^2}}{\pi_1 4}\right) \\
x &= \mu \pm \sqrt{2\sigma^2 \log\left(\frac{\pi_1 4}{\pi_2\sqrt{2\pi\sigma^2}}\right)}
\end{aligned}$$

There is no solutions for $x \notin [-2, 2]$ because the pdf of the multivariate Gaussian is not 0 for any real number. The points -2 and 2 may also be on the decision boundary depending on the exact value of μ, σ, π_1.

(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 part a).

$$x = \pm\sqrt{2\log\left(\frac{36}{11\sqrt{2\pi}}\right)} \approx \pm 0.73$$

The values of x which get classified as Class 2 are if $x \in [-2, -\sqrt{2\log\left(\frac{36}{11\sqrt{2\pi}}\right)}]$ or if $x \in [\sqrt{2\log\left(\frac{36}{11\sqrt{2\pi}}\right)}, 2]$. Otherwise x will get classified as Class 1.



graph of the Bayes decision boundary

(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, and π_1 (instead of using the values that were given in part (b)).

σ, μ can be estimated by calculating the unbiased estimate of the sample mean and sample variance. π_1 can be estimated as the proportion of training data belonging to Class 1.

$$n_1 = \sum_{i:y_i=1} 1 \qquad \text{number of samples of class 1}$$

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

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

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

(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.

From (1),

$$\begin{aligned}
P(Y=1|X=x_0) &= \frac{\pi_1 f_1(X=x_0|Y=1)}{\pi_1 f_1(X=x_0|Y=1)+\pi_2 f_2(X=x_0|Y=2)} \\
&\approx \frac{\hat{\pi_1}\hat{f_1}(X=x_0|Y=1)}{\hat{\pi_1}\hat{f_1}(X=x_0|Y=1)+\hat{\pi_2}\hat{f_2}(X=x_0|Y=2)} \\
&= \frac{\hat{\pi_1}\frac{1}{\sqrt{2\pi\hat{\sigma}^2}}\exp\left(\frac{-(x_0-\hat{\mu})^2}{2\hat{\sigma}^2}\right)}{\hat{\pi_1}\frac{1}{\sqrt{2\pi\hat{\sigma}^2}}\exp\left(\frac{-(x_0-\hat{\mu})^2}{2\hat{\sigma}^2}\right)+(1-\hat{\pi_1})\hat{f_2}(X=x_0|Y=2)} \\
&= \begin{cases} 1 & \text{if } x_0 \notin [-2,2] \\ \dfrac{\hat{\pi_1}\frac{1}{\sqrt{2\pi\hat{\sigma}^2}}\exp\left(\frac{-(x_0-\hat{\mu})^2}{2\hat{\sigma}^2}\right)}{\hat{\pi_1}\frac{1}{\sqrt{2\pi\hat{\sigma}^2}}\exp\left(\frac{-(x_0-\hat{\mu})^2}{2\hat{\sigma}^2}\right)+\frac{(1-\hat{\pi_1})}{4}} & \text{otherwise} \end{cases}
\end{aligned}$$

where $\hat{\pi_1}, \hat{\mu}, \hat{\sigma}$ are as estimated in (c) and only depend on the training observations.

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 $P(Y = 1|X = x)$?
   We don't know the true $P(Y = 1|X = x)$. However, we do know the estimated log-odds of the observation which is given as

$$\begin{aligned}
\log\left(\frac{\hat{P}(Y=1|X=x)}{1-\hat{P}(Y=1|X=x)}\right) &= 0.7 \\
\hat{P}(Y=1|X=x) &= \frac{\exp(0.7)}{1+\exp(0.7)} \approx 0.668
\end{aligned}$$

   (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.
   From the logistic regression model,

$$\begin{aligned}
\log\left(\frac{\hat{P}(Y=1|X=x^*)}{1-\hat{P}(Y=1|X=x^*)}\right) &= \hat{\beta}_0+\hat{\beta}_1(x_1+1)+\hat{\beta}_2(x_2-1)+\hat{\beta}_3(x_3+2)+\hat{\beta}_4x_4^*+\cdots+\hat{\beta}_px_p^* \\
&= \hat{\beta}_1-\hat{\beta}_2+2\hat{\beta}_3+(\hat{\beta}_0+\hat{\beta}_1x_1+\hat{\beta}_2x_2+\hat{\beta}_3x_3+\hat{\beta}_4x_4+\cdots+\hat{\beta}_px_p) \\
&= \hat{\beta}_1-\hat{\beta}_2+2\hat{\beta}_3+0.7 \\
\hat{P}(Y=1|X=x^*) &= \frac{\exp\left(\hat{\beta}_1-\hat{\beta}_2+2\hat{\beta}_3+0.7\right)}{1+\exp\left(\hat{\beta}_1-\hat{\beta}_2+2\hat{\beta}_3+0.7\right)}
\end{aligned}$$

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 $N(\mu_k, \Sigma)$ distribution, for $k = 1, \ldots, K$. 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 $\mu_1, \ldots, \mu_K$.
   The following choices have been made for Σ and $\mu_1, \ldots, \mu_K$:

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

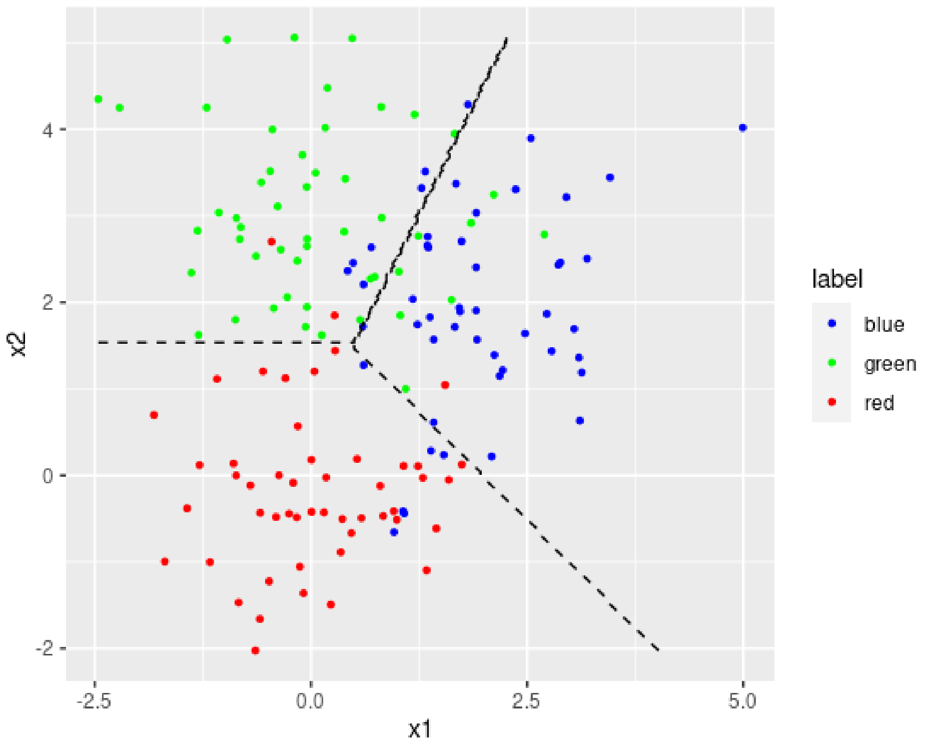
$$\mu_1 = (2, 2) \qquad \text{blue class}$$

$$\mu_2 = (0, 0) \qquad \text{red class}$$

$$\mu_3 = (0, 3) \qquad \text{green class}$$

```r
# 7 a)
library(MASS)
#install.packages("mvtnorm")
library(mvtnorm)
set.seed(42)
mu1 <- c(2, 2)
mu2 <- c(0, 0)
mu3 <- c(0, 3)

sigma = matrix(c((1^2), (1*1*0), (1*1*0), (1^2)), 2)

N = 50
bvn1 <- mvrnorm(N, mu = mu1, Sigma = sigma)
bvn2 <- mvrnorm(N, mu = mu2, Sigma = sigma)
bvn3 <- mvrnorm(N, mu = mu3, Sigma = sigma)

df_blue = data.frame(x1 = bvn1[,1], x2 = bvn1[,2], label = rep("blue", N))
df_red = data.frame(x1 = bvn2[,1], x2 = bvn2[,2], label = rep("red", N))
df_green = data.frame(x1 = bvn3[,1], x2 = bvn3[,2], label = rep("green", N))

X_df = rbind(df_blue, df_red, df_green)
```

(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.

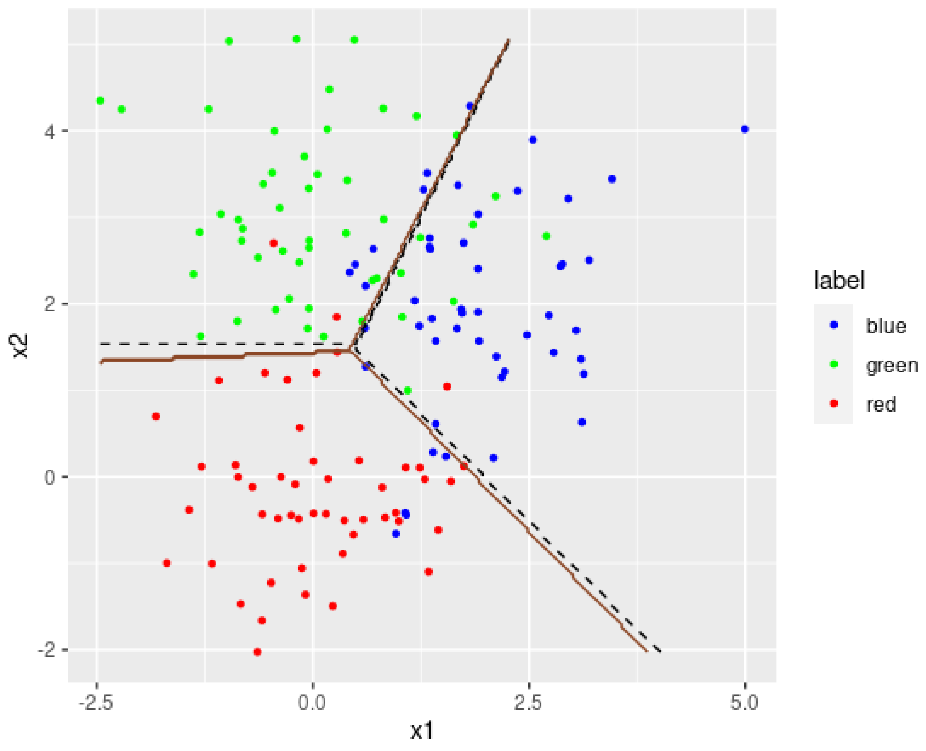


graph of the Bayes decision boundary

```
# 7 b)
min_x1 = min(X_df$x1)
max_x1 = max(X_df$x1)
min_x2 = min(X_df$x2)
max_x2 = max(X_df$x2)

x1_arr <- seq(min_x1, max_x1, length.out=200)
x2_arr <- seq(min_x2, max_x2, length.out=200)

get_bayes_class <- function(x1, x2) {
  f1 = dmvnorm(c(x1, x2), mean = mu1, sigma = sigma)
  f2 = dmvnorm(c(x1, x2), mean = mu2, sigma = sigma)
  f3 = dmvnorm(c(x1, x2), mean = mu3, sigma = sigma)

  return (which.max(c(f1, f2, f3)))
}

bayes_cl_df<-data.frame(x1=rep(x1_arr, length(x2_arr)), x2=rep(x2_arr, each=length(x1_
bayes_cl_df["bayes_class"] = mapply(get_bayes_class, bayes_cl_df$x1, bayes_cl_df$x2)

ggplot(X_df, aes(x1, x2)) + geom_point(aes(color=label), size=1) +
  scale_color_manual(values=c('blue', 'green', 'red')) + coord_fixed() +
  geom_contour(data=bayes_cl_df, aes(x=x1, y=x2, z=bayes_class), colour='black', linet
```

(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)?

The LDA decision boundary is linear and approximately follows the Bayes decision boundary.



graph of the Bayes decision boundary (black) and LDA decision boundary (brown)

```
# 7 c)
get_lda_class <- function(cl){
  if (cl == "red"){
    return (1)
  }
  else if (cl == "blue"){
    return (2)
  }
  else{
    return (3)
  }
}

lda.fit = lda(label ~ x1 + x2, data=X_df)
preds <-predict(lda.fit, bayes_cl_df)
bayes_cl_df["lda_class"] = mapply(get_lda_class, preds$class)

ggplot(X_df, aes(x1, x2)) + geom_point(aes(color=label), size=1) +
  scale_color_manual(values=c('blue', 'green', 'red')) + coord_fixed() +
  geom_contour(data=bayes_cl_df, aes(x=x1, y=x2, z=bayes_class), colour='black', linet
  geom_contour(data=bayes_cl_df, aes(x=x1, y=x2, z=lda_class), colour='sienna4', breal
```

(d) Report the $K \times K$ 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 misclassified).
The confusion matrix is shown below. The training error is around 17%.

```
# 7 d)
train_preds = predict(lda.fit)
table(train_preds$class, X_df$label)
mean(train_preds$class != X_df$label)

        actual
preidcted blue green red
    blue    38     8   2
    green    7    42   3
    red      5     0  45
[1] 0.1666667
```

(e) Generate $n = 50$ test observations in each of the K classes, using the bivariate normal distributions from (a). Report the $K \times K$ 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.
The confusion matrix is shown below. The test error is around 15%.

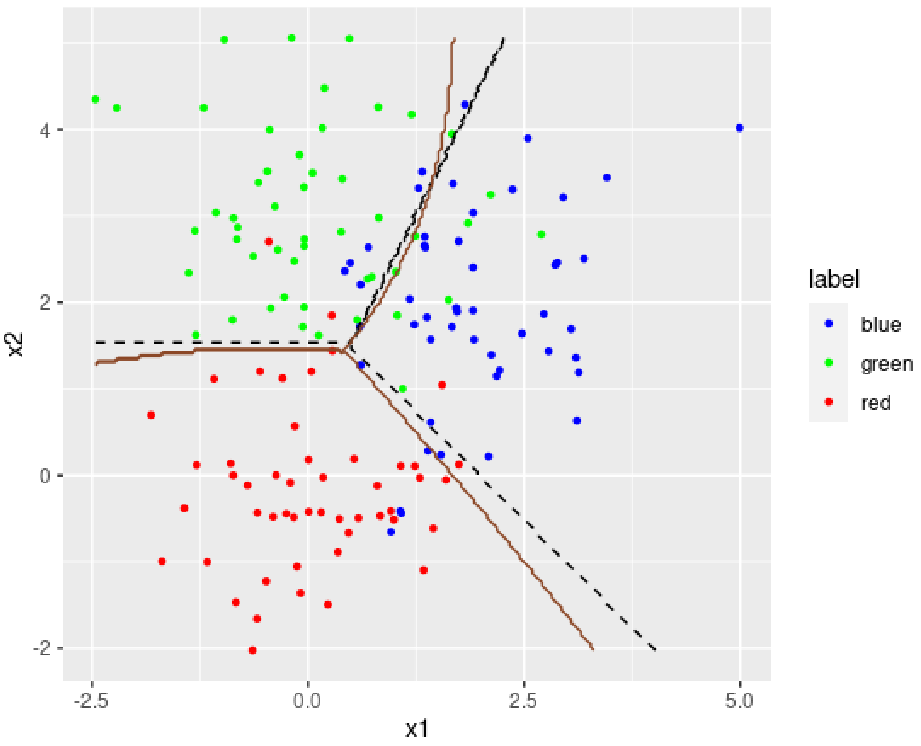
```
# 7 e)
bvn1 <- mvrnorm(N, mu = mu1, Sigma = sigma)
bvn2 <- mvrnorm(N, mu = mu2, Sigma = sigma)
bvn3 <- mvrnorm(N, mu = mu3, Sigma = sigma)

df_blue = data.frame(x1 = bvn1[,1], x2 = bvn1[,2], label = rep("blue", N))
df_red = data.frame(x1 = bvn2[,1], x2 = bvn2[,2], label = rep("red", N))
df_green = data.frame(x1 = bvn3[,1], x2 = bvn3[,2], label = rep("green", N))

X_test_df = rbind(df_blue, df_red, df_green)
test_preds = predict(lda.fit, X_test_df)
table(test_preds$class, X_test_df$label)
mean(test_preds$class != X_test_df$label)

        actual
predicted blue green red
```

```
    blue     42      5   4
    green     7     42   2
    red       1      3  44
[1] 0.1466667
```

(f) Compare your results from (d) and (e), and comment on your findings.
The model performs slightly better on the test set which suggests that it is not overfitting the data and consequently has low variance.

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)?



graph of the Bayes decision boundary (black) and QDA decision boundary (brown)

The QDA decision boundary isn't linear and it deviates from the Bayes decision boundary at some places.

```
# 8 a)
qda.fit = qda(label ~ x1 + x2, data=X_df)
preds <-predict(qda.fit, bayes_cl_df)
bayes_cl_df["qda_class"] = mapply(get_lda_class, preds$class)

ggplot(X_df, aes(x1, x2)) + geom_point(aes(color=label), size=1) +
  scale_color_manual(values=c('blue', 'green', 'red')) + coord_fixed() +
  geom_contour(data=bayes_cl_df, aes(x=x1, y=x2, z=bayes_class), colour='black', linet
  geom_contour(data=bayes_cl_df, aes(x=x1, y=x2, z=qda_class), colour='sienna4', breal
```

(b) Report the $K \times K$ confusion matrix for the QDA model on the training data, as well as the training error.

The confusion matrix is shown below. The training error is around 16%.

```
# 8 b)
train_preds = predict(qda.fit)
table(train_preds$class, X_df$label)
mean(train_preds$class != X_df$label)

        actual
predicted blue green red
    blue    39     8   2
    green    7    42   3
    red      4     0  45
  [1] 0.16
```

(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.)

The confusion matrix is shown below. The test error is around 15%.

```
# 8 c)
test_preds = predict(qda.fit, X_test_df)
table(test_preds$class, X_test_df$label)
mean(test_preds$class != X_test_df$label)

        actual
predicted blue green red
    blue    42     5   3
    green    8    40   2
    red      0     5  45
[1] 0.1533333
```

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

The training and test set error is roughly the same which suggests that the model is not overfitting the data.

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

The QDA model had slightly smaller training error compared to LDA. This could be due to QDA being a more flexible model which allows it to better fit the training data.

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

The LDA model had a slightly smaller test error compared to QDA. This could be due to QDA overfitting the training set which causes it to not generalize well to the test set.

9. We have seen in class that the least squares regression estimator involves finding the coefficients β_0, β_1, ..., β_p that minimize the quantity

$$\sum_{i=1}^{n}(y_i - (\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}))^2$$

By contrast, the ridge regression estimator (which we will discuss in Chapter 6) involves finding the coefficients that minimize

$$\sum_{i=1}^{n}(y_i - (\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}))^2 + \lambda(\beta_1^2 + \cdots + \beta_p^2)$$

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 $\hat{\beta}_0, \ldots, \hat{\beta}_p$.

The term that we are minimizing can be written as the following in the matrix notation.

$$\begin{aligned}
L(\overrightarrow{\beta}) &= \|\mathbf{y} - \mathbf{X}\overrightarrow{\beta}\|_{\mathbf{2}}^{\mathbf{2}} + \lambda\|\overrightarrow{\beta}\|_{\mathbf{2}}^{\mathbf{2}} \\
\frac{\partial L(\overrightarrow{\beta})}{\partial \overrightarrow{\beta}} &= \frac{\partial(\|\mathbf{y} - \mathbf{X}\overrightarrow{\beta}\|_{\mathbf{2}}^{\mathbf{2}} + \lambda\|\overrightarrow{\beta}\|_{\mathbf{2}}^{\mathbf{2}})}{\partial \overrightarrow{\beta}} \\
&= \frac{\partial(\mathbf{y^T y} - \mathbf{y^T X}\overrightarrow{\beta} - (\mathbf{X}\overrightarrow{\beta})^{\mathbf{T}}\mathbf{y} + (\mathbf{X}\overrightarrow{\beta})^{\mathbf{T}}(\mathbf{X}\overrightarrow{\beta}) + \lambda\|\overrightarrow{\beta}\|_{\mathbf{2}}^{\mathbf{2}})}{\partial \overrightarrow{\beta}} \\
&= -\frac{\partial(\mathbf{y^T X}\overrightarrow{\beta})}{\partial \overrightarrow{\beta}} - \frac{\partial(\mathbf{X}\overrightarrow{\beta})^{\mathbf{T}}\mathbf{y})}{\partial \overrightarrow{\beta}} + \frac{\partial(\mathbf{X}\overrightarrow{\beta})^{\mathbf{T}}(\mathbf{X}\overrightarrow{\beta}))}{\partial \overrightarrow{\beta}} + \frac{\partial(\lambda\|\overrightarrow{\beta}\|_2^2)}{\partial \overrightarrow{\beta}} \\
&= -2\frac{\partial(\mathbf{y^T X}\overrightarrow{\beta})}{\partial \overrightarrow{\beta}} + \frac{\partial(\overrightarrow{\beta}^T\mathbf{X^T X}\overrightarrow{\beta})}{\partial \overrightarrow{\beta}} + \frac{\partial(\lambda\|\overrightarrow{\beta}\|_2^2)}{\partial \overrightarrow{\beta}} \\
0 &= -2\mathbf{X^T y} + \mathbf{2X^T X}\overrightarrow{\beta} + \mathbf{2}\lambda\overrightarrow{\beta} \\
\implies \overrightarrow{\beta} &= (\mathbf{X^T X} + \lambda\mathbf{I})^{\mathbf{-1}}\mathbf{X^T y}
\end{aligned}$$