Homework 4

Phoebe Parrish

2/29/2020

Question 1

In this exercise, you will generate simulated data, and will use this data to perform the lasso.

(A) Use the `rnorm()` function to generate a predictor X of length $n = 30$, and a noise vector ϵ of length $n = 30$.

(B) Generate a response vector Y of length $n = 30$ according to the model:

$$Y = 3 - 2X + 3 * X^2 + \epsilon$$

(C) Fit a lasso model to the data, using $X, X^2, ..., X^7$ in order to predict Y.

- Make a plot that displays the value of each coefficient, as a function of λ. You can display λ on the x-axis and "Coefficient Value" on the y-axis.
- Use cross-validation to select the tuning parameter. What tuning parameter value do you choose? Make a plot to justify your choice. Your plot could display "Estimated Test Error" on the y-axis and λ on the x-axis (or it could display other quantities of your choice).
- Fit a lasso model to all n observations, using the tuning parameter value selected in the previous sub-problem. Write out the fitted model. Comment on the fitted model.

(D) Now generate 1000 new observations as in 1(A) and 1(B). Apply the final fitted model from 1(C) to these new observations. What is the mean squared error?

(A)

```
set.seed(123)

X <- rnorm(n=30, mean=2, sd=0.5)
epsilon <- rnorm(n=30)
```

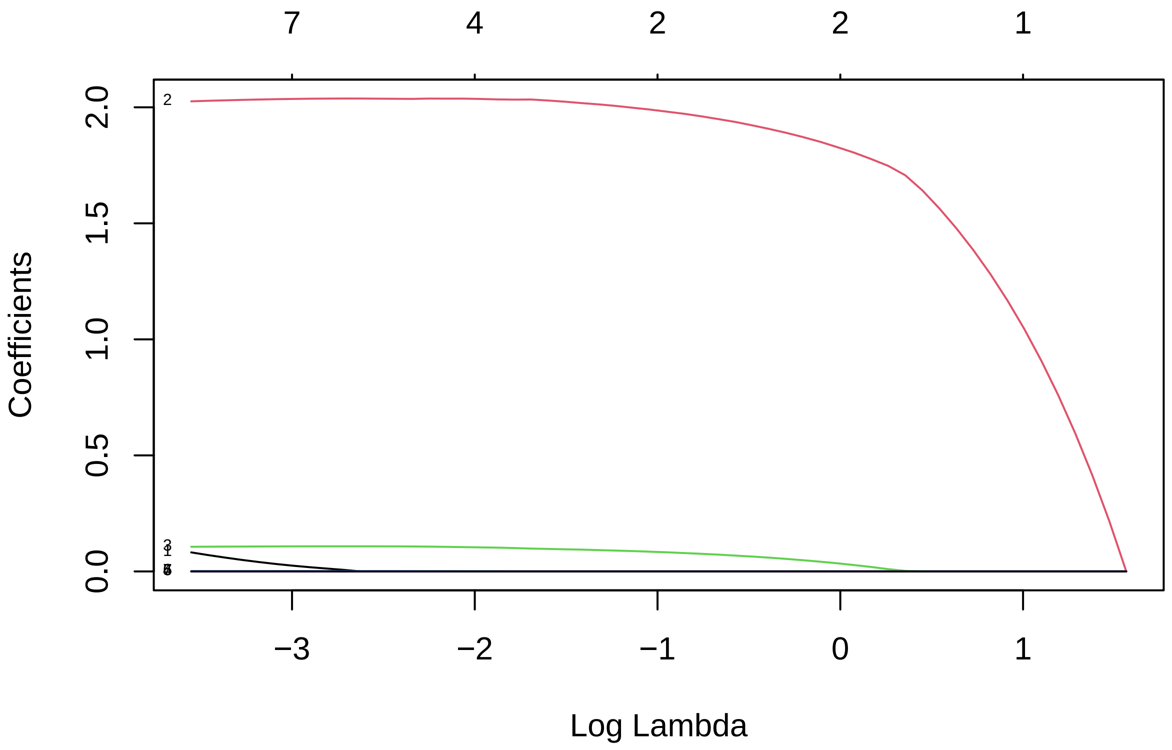
(B)

```
set.seed(123)

Y <- 3 - 2*X + 3*(X^2) + epsilon

# save all variables in a dataframe
df.lasso <- tibble(Y=Y, X=X, epsilon=epsilon)
```

(C)

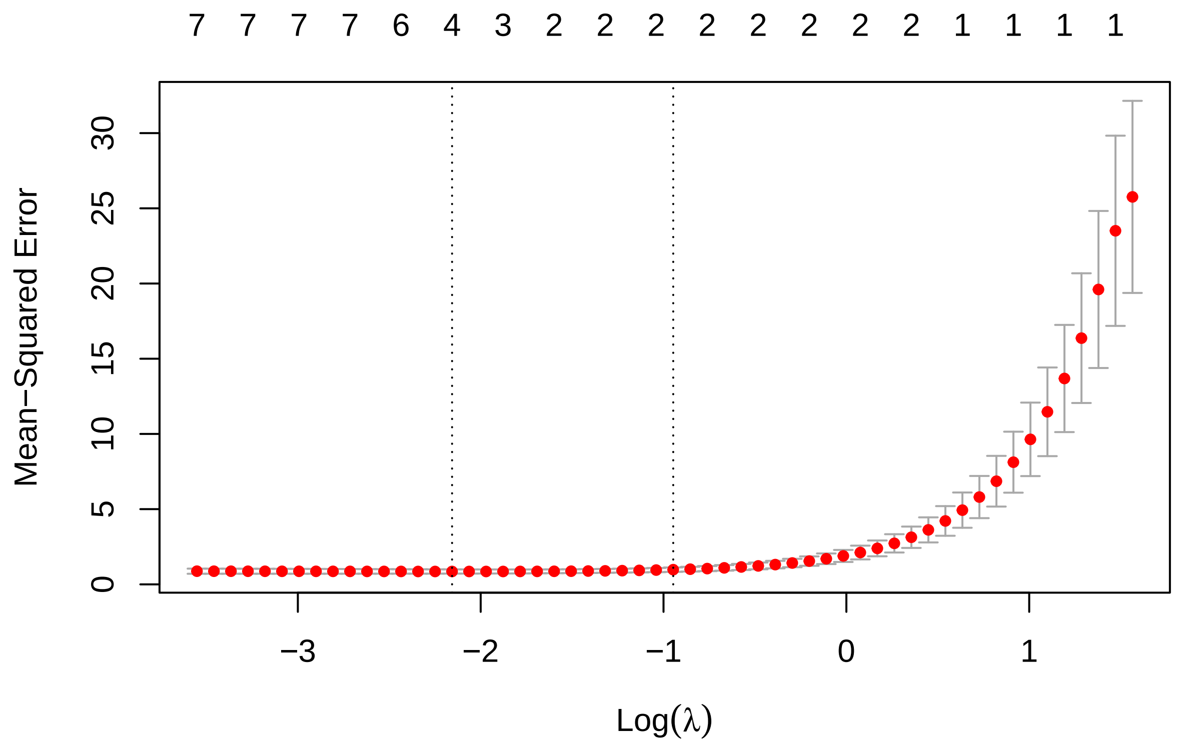
(i)

```
set.seed(123)

df.X.poly <- tibble(X1=X, X2=X^2, X3=X^3, X4=X^4, X5=X^5, X6=X^6, X7=X^7)

glmnet.X <- as.matrix(df.X.poly)

lasso.fit <- glmnet(x=glmnet.X, y=Y, a=1)
plot(lasso.fit, xvar="lambda", label=TRUE)
```



(ii)

```
set.seed(123)

cv.lasso.fit <- cv.glmnet(x=glmnet.X, y=Y, alpha=1)

plot(cv.lasso.fit)
```



```r
lambda.min.lasso <- cv.lasso.fit$lambda.min
cat("lambda.min =", lambda.min.lasso)
cat("\nlog(lambda.min) =", log10(lambda.min.lasso))

lasso.error.min <- cv.lasso.fit$cvm[cv.lasso.fit$lambda == cv.lasso.fit$lambda.min]
cat("\n\nTest error at lambda.min =", lasso.error.min)
```

```
## lambda.min = 0.1157389
## log(lambda.min) = -0.9365206
##
## Test error at lambda.min = 0.8550821
```

(iii)

```r
set.seed(123)

lasso.fit2 <- glmnet(x=glmnet.X, y=Y, lambda=lambda.min.lasso, a=1)

coef(lasso.fit2)
```

```
## 8 x 1 sparse Matrix of class "dgCMatrix"
##                     s0
## (Intercept) 1.33104606
## X1          1.46336270
## X2          1.33153111
## X3          0.18322208
## X4          0.01133323
## X5          .
## X6          .
## X7          .
```

$$Y = 1.33104606 + 1.46336269X + 1.33153111X^2 + 0.18322208X^3 + 0.01133323X^4$$

Lasso does feature selection, so here our X^5, X^6, X^7 features are removed from the fitted model. This means we are less likely to overfit to our training data by including features that are not as important, and also means that the model is more interpretable. In this case, we see that X^1 and X^2 have high β coefficient values, suggesting that they are useful for predicting Y. This makes sense given that we generated Y using X and X^2.

(D)

```r
set.seed(123)

X.new <- rnorm(n=1000, mean=2, sd=0.5)
epsilon.new <- rnorm(n=1000)
df.X.new.poly <- tibble(X1=X.new, X2=X.new^2, X3=X.new^3, X4=X.new^4, X5=X.new^5,
                        X6=X.new^6, X7=X.new^7)
glmnet.X.new <- as.matrix(df.X.new.poly)

Y.new <- 3 - 2*X.new + 3*(X.new^2) + epsilon.new

Yhat.pred <- predict(lasso.fit2, newx=glmnet.X.new, s=lambda.min.lasso)

lasso.pred.df <- tibble(Y=Y.new, Y.hat=Yhat.pred) %>% mutate(diff.sq = (Y-Y.hat)^2)
mse.lasso <- mean(lasso.pred.df$diff.sq)

cat("Lasso MSE =", mse.lasso)
```

```
## Lasso MSE = 1.129267
```

Question 2

In this exercise, you will apply ridge regression to the data that you generated in 1(A), 1(B), and 1(D).

(A) Fit a ridge regression model to the data, using $X, X^2, ..., X^7$ in order to predict Y.

- Make a plot that displays the value of each coefficient, as a function of λ. You can display λ on the x-axis and "Coefficient Value" on the y-axis.
- Use cross-validation to select the tuning parameter. What tuning parameter value do you choose? Make a plot to justify your choice. Your plot could display "Estimated Test Error" on the y-axis and λ on the x-axis (or it could display other quantities of your choice).
- Fit a ridge regression model to all n observations, using the tuning parameter value selected in the previous sub-problem. Write out the fitted model. Comment on the fitted model.

(B) Apply the final fitted model from 2(A) to the observations generated in 1(D). What is your mean squared error?

(C) Now fit a least squares model to the data generated in 1(A) and 1(B), using $X, X^2, ..., X^7$ in order to predict Y. Then apply this fitted model to the 1000 new observations generated in 1(D). What mean squared error do you get?

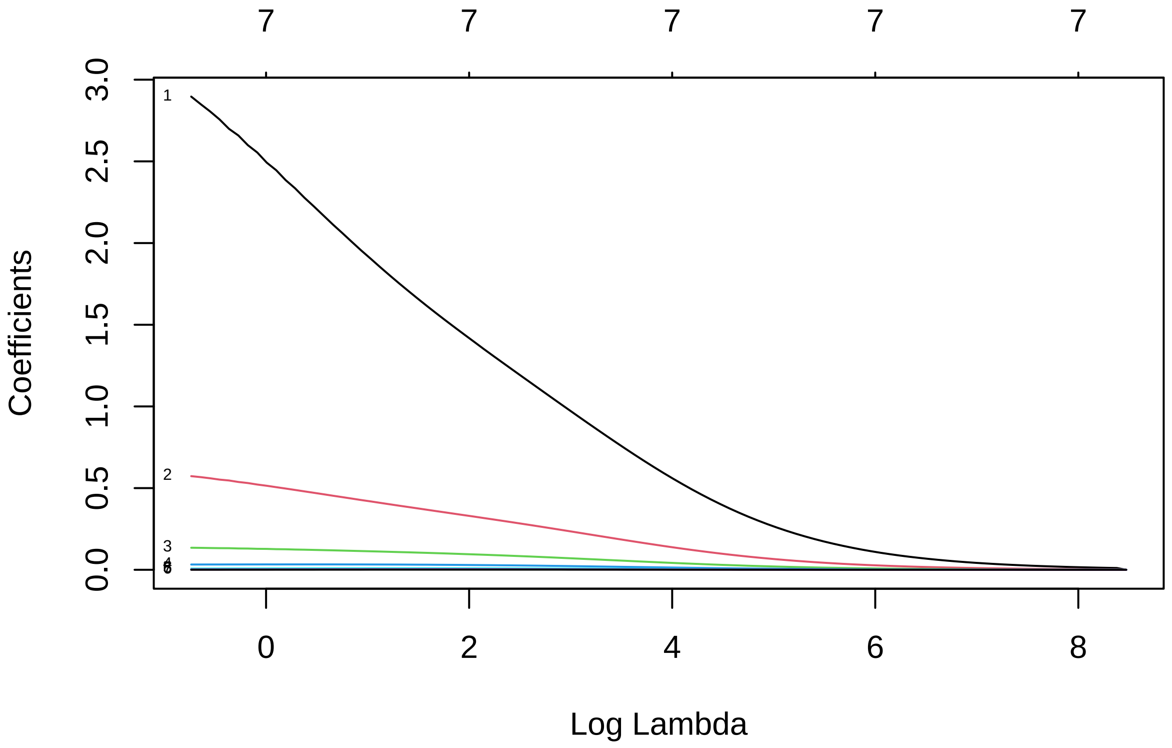
(D) Given your answers to 1(D) and 2(B) and 2(C), which is a better choice on this data: ridge regression, the lasso, or least squares?

(A)

```r
set.seed(123)

ridge.fit <- glmnet(x=glmnet.X, y=Y, a=0)
```

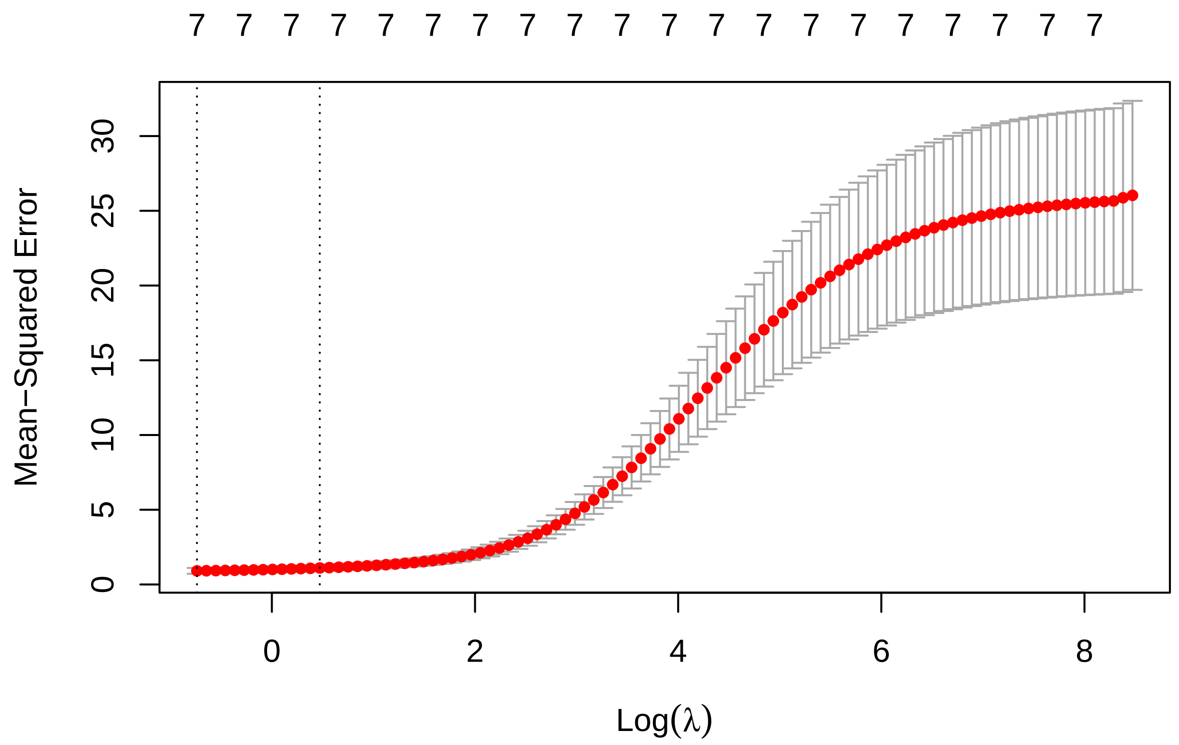
(i)

```
set.seed(123)

plot(ridge.fit, xvar="lambda", label=TRUE)
```



(ii)

```
set.seed(123)

cv.ridge.fit <- cv.glmnet(x=glmnet.X, y=Y, alpha=0)

plot(cv.ridge.fit)
```



```
lambda.min.ridge <- cv.ridge.fit$lambda.min
cat("lambda.min =", lambda.min.ridge)
cat("\nlog(lambda.min) =", log10(lambda.min.ridge))

ridge.error.min <- cv.ridge.fit$cvm[cv.ridge.fit$lambda == cv.ridge.fit$lambda.min]
cat("\n\nTest error at lambda.min =", ridge.error.min)
```

```
## lambda.min = 0.4782346
## log(lambda.min) = -0.320359
##
## Test error at lambda.min = 0.9112119
```

(iii)

```
set.seed(123)

ridge.fit2 <- glmnet(x=glmnet.X, y=Y, lambda=lambda.min.ridge, a=0)

coef(ridge.fit2)
```

```
## 8 x 1 sparse Matrix of class "dgCMatrix"
##                       s0
## (Intercept) 1.0131669074
## X1          2.8945054996
## X2          0.5716611385
## X3          0.1338269816
## X4          0.0321888349
## X5          0.0075628021
## X6          0.0015955537
## X7          0.0002344543
```

$Y = 1.0131669158 + 2.8945054953X + 0.5716611381X^2 + 0.1338269816X^3 + 0.0321888349X^4 + 0.0075628021X^5 +$

$0.0015955537X^5 + 0.0015955537X^6 + 0.0002344543X^7$

Ridge regression does *not* perform feature selection, so all of the features are still included in our model. Here we have many features with low beta coefficient values, which are likely not very important in predicting the response variable Y. This contrasts directly with our lasso model, because in lasso the X^5, X^6, and X^7 coefficients (which have the lowest β coefficient values here) were removed.

(B)

```r
set.seed(123)

Yhat.pred.ridge <- predict(ridge.fit2, newx=glmnet.X.new, s=lambda.min.ridge)

ridge.pred.df <- tibble(Y=Y.new, Y.hat=Yhat.pred.ridge) %>% mutate(diff.sq = (Y-Y.hat)^2)
mse.ridge <- mean(ridge.pred.df$diff.sq)

cat("Ridge MSE =", mse.ridge)
```

```
## Ridge MSE = 1.247067
```

(C)

```r
set.seed(123)

df.sim <- bind_cols(tibble(Y), df.X.poly)

lm.fit <- lm(Y ~ ., data=df.sim)

lm.pred <- predict(lm.fit, df.X.new.poly)

df.sim.pred <- bind_cols(Y=Y.new, Y.hat=lm.pred) %>% mutate(diff=(Y-Y.hat)^2)
mse.lm <- mean(df.sim.pred$diff)

cat("LSLM MSE =", mse.lm)
```

```
## LSLM MSE = 755.1114
```

(D)

```r
mse.compare.df <- tibble(model=c("lasso", "ridge", "LSLM"),
                         MSE=c(mse.lasso, mse.ridge, mse.lm))
kable(mse.compare.df)
```

model	MSE
lasso	1.129267
ridge	1.247067
LSLM	755.111443

In this case, I would go with the lasso model because it gives me the lowest test error.

Question 3

In this problem, we will simulate some data, and we'll compare the results that we get using least squares linear regression, and using a regression tree. Let $n = 100$ and $p = 2$.

(A) Generate $n = 100$ observations according to the linear model:

$$Y = 1 + 2X_1 + 3X_2 + \epsilon$$

You can generate X_1, X_2, and ϵ independently from a $N(0, 1)$ distribution.

(B) Make a plot of the data. One axis of your plot should represent X_1, one axis should represent X_2, and the color of each point should represent the value of Y. You can use a command like this one: `plot(x1, x2, col=rainbow(200)[rank(y)], pch=15)`.

(C) Do you expect least squares linear regression or a regression tree to perform better on this data, in terms of test error? Explain your answer.

(D) Fit a least squares linear model to the data, and fit a regression tree to the data. (Be sure to prune your tree, if appropriate!) Report the fitted model for the former, and display the tree for the latter. Make sure that the nodes in your regression tree are labeled appropriately.

(E) Repeat the plot from (B), but this time display the partitions of feature space corresponding to the tree from (D). Furthermore, label each region with the predicted response value in this region. The predicted response value in each region should be displayed, and the color of each observation should reflect the corresponding response value, as mentioned in (B).

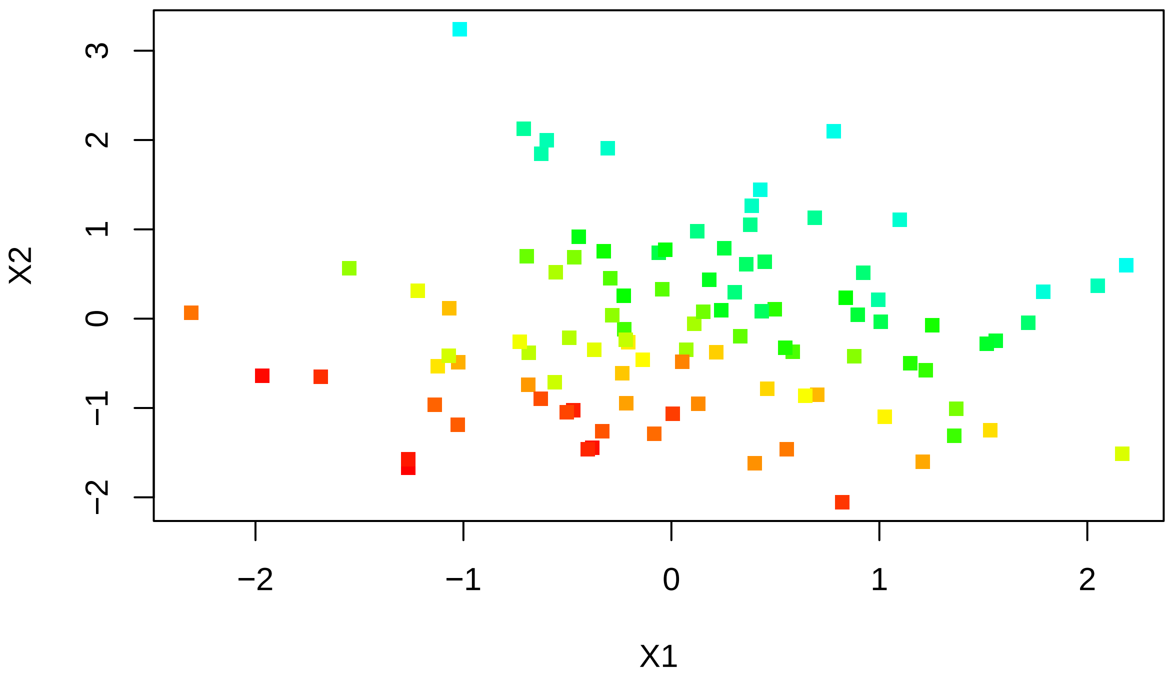
(F) Generate 1000 test observations, and report the test error for both of the models that you fit in (D). Comment on your results.

(A)

```
set.seed(123)

X1 <- rnorm(100)
X2 <- rnorm(100)
epsilon <- rnorm(100)

Y <- 1 + 2*X1 + 3*X2 + epsilon

df.sim.new <- tibble(Y=Y, X1=X1, X2=X2)
```

(B)

```
set.seed(123)

plot(df.sim.new$X1, df.sim.new$X2, col=rainbow(200)[rank(df.sim.new$Y)],
     pch=15, xlab="X1", ylab="X2")
```



(C) I would expect a least squares linear model to perform fairly well on these data, since by eye it looks like you could separate the observations linearly.

(D)
Least squares linear model:

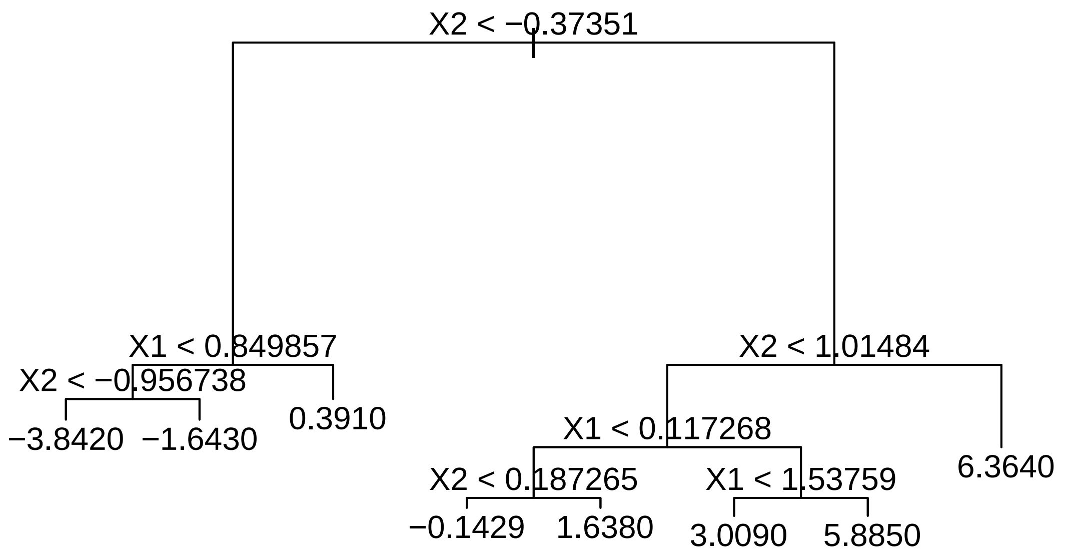
```
lm.fit.new <- lm(Y ~ ., data=df.sim.new)

summary(lm.fit.new)
```

```
##
## Call:
## lm(formula = Y ~ ., data = df.sim.new)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.8730 -0.6607 -0.1245  0.6214  2.0798
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.13507    0.09614   11.81   <2e-16 ***
## X1           1.86683    0.10487   17.80   <2e-16 ***
## X2           3.02381    0.09899   30.55   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9513 on 97 degrees of freedom
## Multiple R-squared:  0.9252, Adjusted R-squared:  0.9236
## F-statistic: 599.5 on 2 and 97 DF,  p-value: < 2.2e-16
```

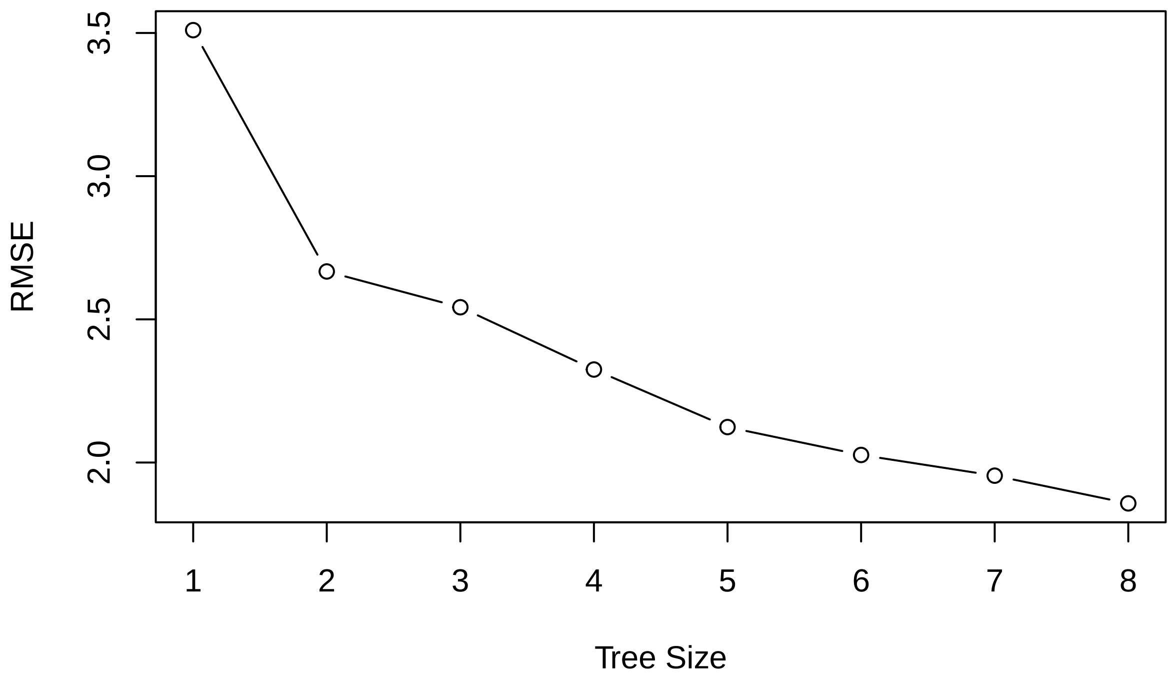
$$Y = 1.13507 + 1.86683X_1 + 3.02381X_2$$

Regression tree:

```r
set.seed(123)

sim.reg.tree <- tree(Y ~ ., data=df.sim.new)

summary(sim.reg.tree)

plot(sim.reg.tree)
text(sim.reg.tree, pretty=0)
```



```r
cv.sim.reg.tree <- cv.tree(sim.reg.tree)
plot(cv.sim.reg.tree$size, sqrt(cv.sim.reg.tree$dev/nrow(df.sim.new)), type='b',
     xlab="Tree Size", ylab="RMSE")
```



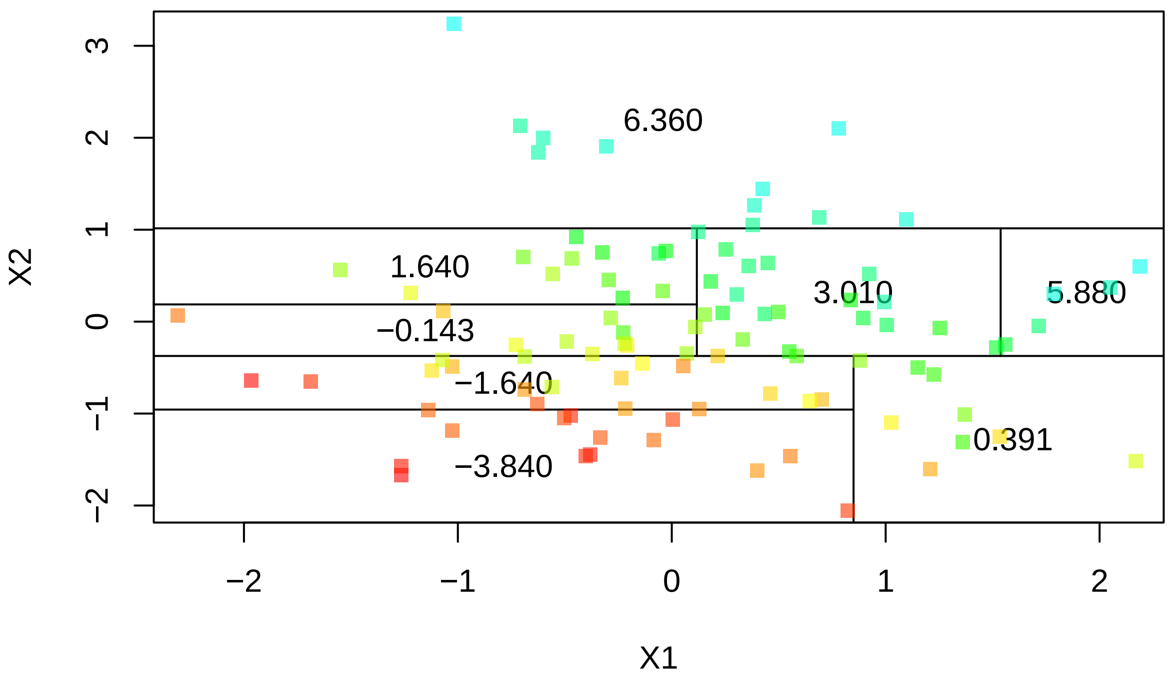
```
##
## Regression tree:
```

```
## tree(formula = Y ~ ., data = df.sim.new)
## Number of terminal nodes:  8
## Residual mean deviance:  1.955 = 179.8 / 92
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -3.9350 -0.8412  0.1152  0.0000  0.8922  3.6220
```

Here I did not prune my tree because the lowest cross-validation error was achieved with 8 branches.

(E)

```
partition.tree(sim.reg.tree, label="yval", ordvars=c("X1", "X2"))
points(df.sim.new$X1, df.sim.new$X2, col=rainbow(200, alpha=0.6)[rank(df.sim.new$Y)],
       pch=15, xlab="X1", ylab="X2")
```



(F)

```
set.seed(123)

X1.test <- rnorm(1000)
X2.test <- rnorm(1000)
epsilon.test <- rnorm(1000)

Y.test <- 1 + 2*X1.test + 3*X2.test + epsilon

df.sim.new.test <- tibble(Y=Y.test, X1=X1.test, X2=X2.test)

yhat.reg.tree <- predict(sim.reg.tree, newdata=df.sim.new.test)

mse.tree <- mean((yhat.reg.tree-df.sim.new.test$Y)^2)
cat("Tree MSE =", mse.tree)

df.sim.new.pred <- predict(lm.fit.new, df.sim.new.test)

df.sim.new.pred <- bind_cols(Y=Y.test, Y.hat=df.sim.new.pred) %>% mutate(diff=(Y-Y.hat)^2)
```

```r
mse.lm.new <- mean(df.sim.new.pred$diff)

cat("\nLSLM MSE =", mse.lm.new)
```

```
## Tree MSE = 4.347902
## LSLM MSE = 0.9278613
```

Question 4

Repeat the previous problem, but this time generate data as follows:

$$Y = 2 + 3I_{(X1<0)} + 0.5I_{(X_1\geq 0; X_2<0.5)} - 2I_{(X_1\geq 0; X_2\geq 0.5)} + \epsilon$$

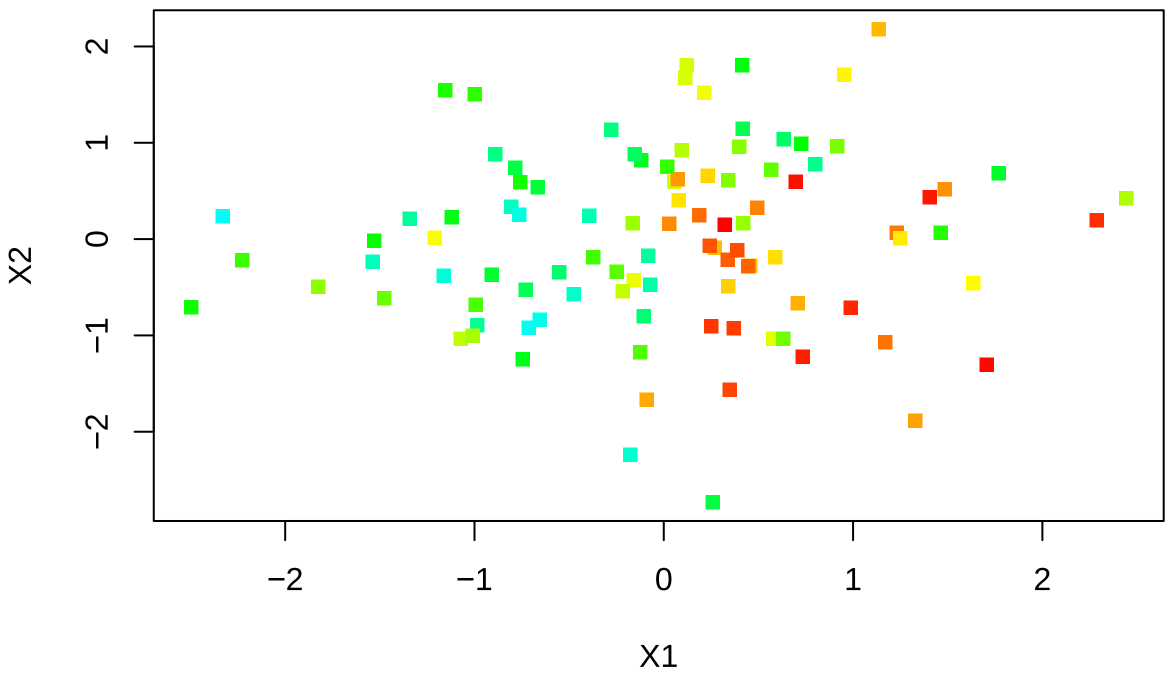
Here, $I_{(A)}$ is an indicator variable that equals 1 if the event A holds, and equals 0 otherwise.

(A)

```r
set.seed(321)

X1.4 <- rnorm(100)
X2.4 <- rnorm(100)
epsilon.4 <- rnorm(100)

dummy1 <- ifelse(X1.4 < 0, 1, 0)
dummy2 <- ifelse(X1.4 >= 0 & X2.4 < 0.5, 1, 0)
dummy3 <- ifelse(X1.4 >= 0 & X2.4 >= 0.5, 1, 0)

Y.4 <- 2 + 3*dummy1 + 0.5*dummy2 + 2*dummy3 + epsilon.4

df.sim.4 <- tibble(Y=Y.4, X1=X1.4, X2=X2.4)
```

(B)

```r
set.seed(321)

plot(df.sim.4$X1, df.sim.4$X2, col=rainbow(200)[rank(df.sim.4$Y)], pch=15,
     xlab="X1", ylab="X2")
```



(C) I expect a tree to do better on these data because they are good at classification.

(D)

Linear model:

```
lm.fit.4 <- lm(Y ~ ., data=df.sim.4)
summary(lm.fit.4)
```

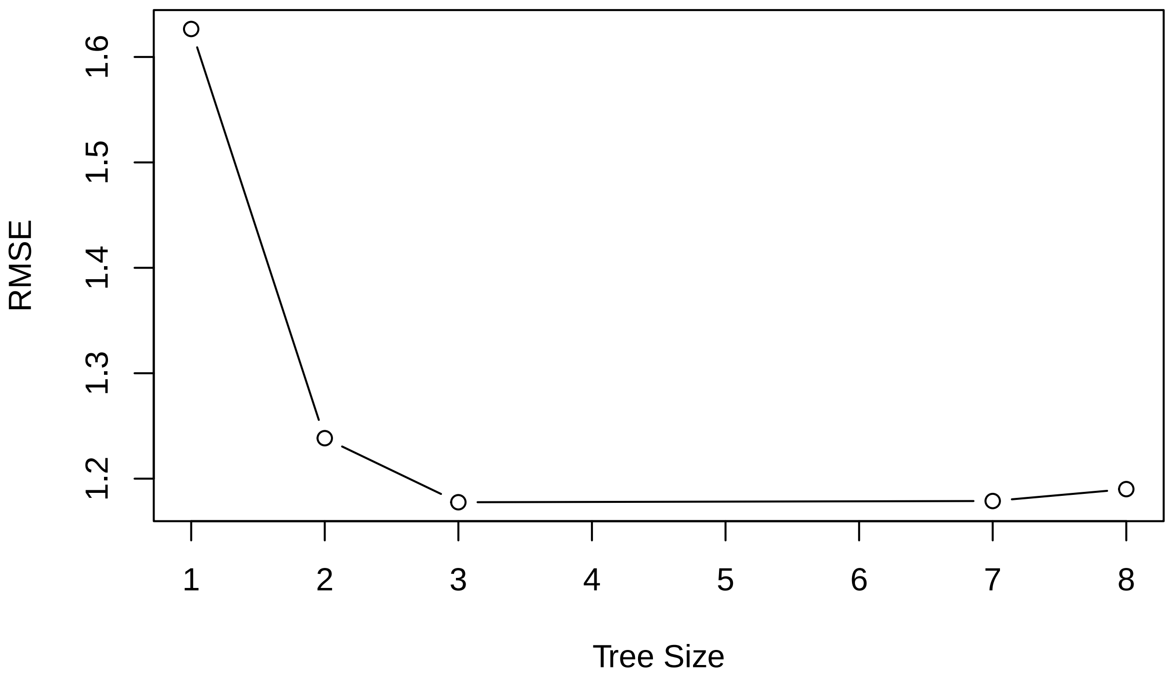
```
##
## Call:
## lm(formula = Y ~ ., data = df.sim.4)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7635 -0.9462 -0.1385  0.9269  3.0026
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   4.1276     0.1345  30.685  < 2e-16 ***
## X1           -0.9343     0.1421  -6.577 2.44e-09 ***
## X2            0.1940     0.1484   1.307    0.194
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.345 on 97 degrees of freedom
## Multiple R-squared:  0.3108, Adjusted R-squared:  0.2966
## F-statistic: 21.87 on 2 and 97 DF,  p-value: 1.448e-08
```

$$Y = 4.1276 + -0.9343X_1 + 0.1940X_2$$

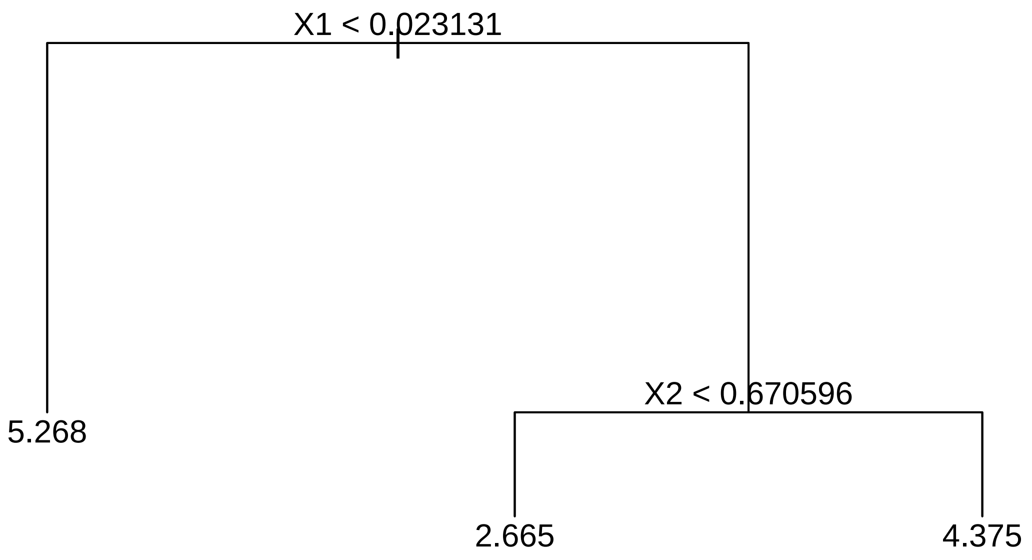
Tree:

```
set.seed(321)
```

```r
sim.class.tree <- tree(Y ~ ., data=df.sim.4)

summary(sim.class.tree)

cv.sim.class.tree <- cv.tree(sim.class.tree)
plot(cv.sim.class.tree$size, sqrt(cv.sim.class.tree$dev/nrow(df.sim.new)), type='b',
     xlab="Tree Size", ylab="RMSE")
```



```r
pruned.class.tree <- prune.tree(sim.class.tree, best=3)

plot(pruned.class.tree)
text(pruned.class.tree, pretty=0)
```



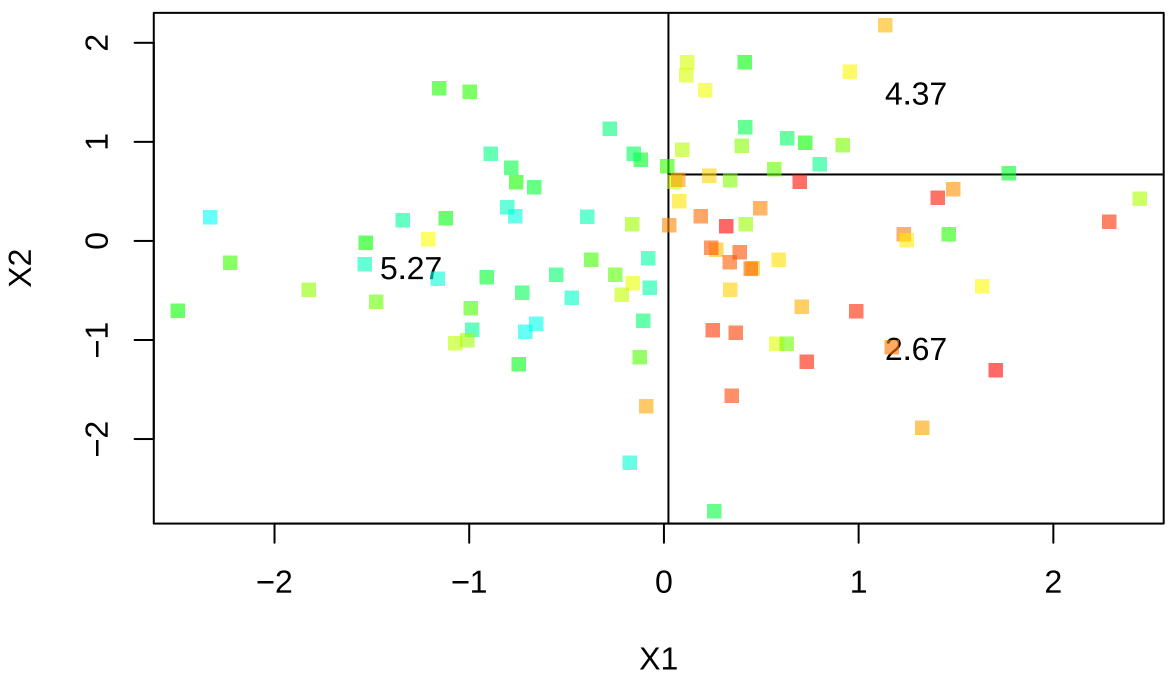
```
##
## Regression tree:
```

```
## tree(formula = Y ~ ., data = df.sim.4)
## Number of terminal nodes:  8
## Residual mean deviance:  0.953 = 87.68 / 92
## Distribution of residuals:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -1.7010 -0.7234 -0.1358  0.0000  0.6125  2.6510
```

In this case, I chose to prune my tree from 8 terminal nodes to 3 because the lowest cross-validation error rate occured at a tree size of 3.

(E)

```
partition.tree(pruned.class.tree, label="yval", ordvars=c("X1", "X2"))
points(df.sim.4$X1, df.sim.4$X2, col=rainbow(200, alpha=0.6)[rank(df.sim.4$Y)],
       pch=15, xlab="X1", ylab="X2")
```



(F)

```
set.seed(321)

X1.4.test <- rnorm(1000)
X2.4.test <- rnorm(1000)
epsilon.4.test <- rnorm(1000)

Y.4.test <- 1 + 2*X1.test + 3*X2.test + epsilon

df.sim.4.test <- tibble(Y=Y.4.test, X1=X1.4.test, X2=X2.4.test)

yhat.class.tree <- predict(pruned.class.tree, newdata=df.sim.4.test)

mse.class.tree <- mean((yhat.class.tree-df.sim.4.test$Y)^2)
cat("Tree MSE =", mse.class.tree)

df.sim.4.pred <- predict(lm.fit.4, df.sim.4.test)
df.sim.4.pred <- bind_cols(Y=Y.4.test, Y.hat=df.sim.4.pred) %>% mutate(diff=(Y-Y.hat)^2)
mse.lm.4 <- mean(df.sim.4.pred$diff)

cat("\nLSLM MSE =", mse.lm.4)
```

```
## Tree MSE = 25.05518
## LSLM MSE = 23.90241
```

Here, it looks like I got slightly lower mean squared error for the predictions from my least squares linear model compared to the predictions from my classification tree model. However, the errors are very close. In a case like this, I would likely choose to go with a decision tree model because the error rates are similar and a tree easier to interpret and explain to others.