DATA 558: HW Assignment 3

Hriday Baghar

May 11, 2022

Instructions:

You may discuss the homework problems in small groups, but you must write up the final solutions and code yourself. Please turn in your code for the problems that involve coding. However, code without written answers will receive no credit. To receive credit, you must explain your answers and show your work. All plots should be appropriately labeled and legible, with axis labels, legends, etc., as needed.

On this assignment, some of the problems involve random number generation. Be sure to set a random seed (using the command `set.seed()`*) before you begin.*

```
library(ISLR2)
library(glmnet)
library(ggplot2)
library(boot)
library(dplyr)
library(glmnet)
```

1. In this problem, we'll see a (very!!) simple simulated example where a least squares linear model is "too flexible".

(a) First, generate some data with n = 100 and p = 10,000 features, and a quantitative response, using the following R commands:

$$\texttt{y} < -\texttt{rnorm}(\texttt{100})$$

$$\texttt{x} < -\texttt{matrix}(\texttt{rnorm}(\texttt{10000} * \texttt{100}), \texttt{ncol} = \texttt{10000})$$

Write out an expression for the model corresponding to this data generation procedure. For instance, it might look something like $Y = 2X_1 + 3X_2 + \varepsilon, \varepsilon \sim N(0, 1)$.

```
set.seed(42) # the answer to the ultimate question of life, the universe, and everything

y <- rnorm(100)
x <- matrix(rnorm(10000*100), ncol = 10000)
```

The model we are building here has no features and is pure noise. Hence the expression looks like:

$$Y = \varepsilon, \quad \varepsilon \sim N(0, 1)$$

(b) What is the value of the irreducible error?

$Var(\varepsilon) = 1$ is the value of the irreducible error.

(c) Consider a very simple model-fitting procedure that just predicts 0 for every observation. That is, $\hat{f}(x) = 0$ for all x.

i. What is the bias of this procedure?

For a single observation X_0:

$$\begin{aligned} Bias &= E[f(X_0) - \hat{f}(X_0)] \\ &= f(X_0) \\ &= \varepsilon_0 \end{aligned} \tag{1}$$

To calculate the overall bias we take the mean of bias for all points in the data. Since we know ε is a standard normal random variable, $E[\varepsilon] = 0$. Therefore, Bias of this procedure is 0.

ii. What is the variance of this procedure?

$$Var(f(\hat{X}_0)) = E[\hat{f}(X_0)^2] - E[\hat{f}(X_0)]^2 = 0$$

Because $\hat{f}$ always predicts 0.

iii. What is the expected prediction error of this procedure?

$$\begin{aligned} MSE &= Var(\varepsilon) + Bias^2(\hat{f}(X_0)) + Var(\hat{f}(X_0)) \\ &= Var(\varepsilon) + 0 + 0 \\ &= 1 \end{aligned} \tag{2}$$

iv. Use the validation set approach to estimate the test error of this procedure. What answer do you get?

```
set.seed(42)

rows <- sample(1:100, size = 70, replace = FALSE)

x.train <- x[rows,]
x.val <- x[-rows,]

y.train <- y[rows]
y.val <- y[-rows]

mean(y.val^2)
```

```
## [1] 1.288504
```

v. Comment on your answers to (iii) and (iv). Do your answers agree with each other? Explain.

We see that the MSE in (iv) is approximately close to the true value of 1 that we calculate in (iii). The difference in values can be attributed to the small sample which we use to estimate the MSE.

(d) Now use the validation set approach to estimate the test error of a least squares linear model using $X_1, ..., X_{10,000}$ to predict Y . What is the estimated test error?

Hint: If you fit a least squares linear model to predict Y using $X_1, ..., X_p$ where $p \geq n$, then only the first $n - 1$ coefficients will be assigned values. The rest will show up as NA because those coefficients aren't needed to obtain a perfect (i.e. zero) residual sum of squares on the training data. You can see all of the coefficient values by applying the `coef()` command to the output of the linear model.

```
train <- data.frame(x.train, y=y.train)
val <- data.frame(x.val, y=y.val)

model.1 <- lm(y ~ ., data = train)
preds <- predict(model.1, newdata = val)
```

```
## Warning in predict.lm(model.1, newdata = val): prediction from a rank-deficient
## fit may be misleading
```

```
mean((val[,"y"] - preds)^2)
```

```
## [1] 56.66237
```

Estimated test error is 56.6623727.

(e) Comment on your answers to (c) and (d). Which of the two procedures has a smaller estimated test error? higher bias? higher variance? In answering this question, be sure to think carefully about how the data were generated.

The procedure of estimating test error in (c) has a smaller test error since we use the true form of f, moreover our estimated $\hat{f}$ is also based on properties of the standard normal distribution (mean value of $\hat{f}$ is 0).

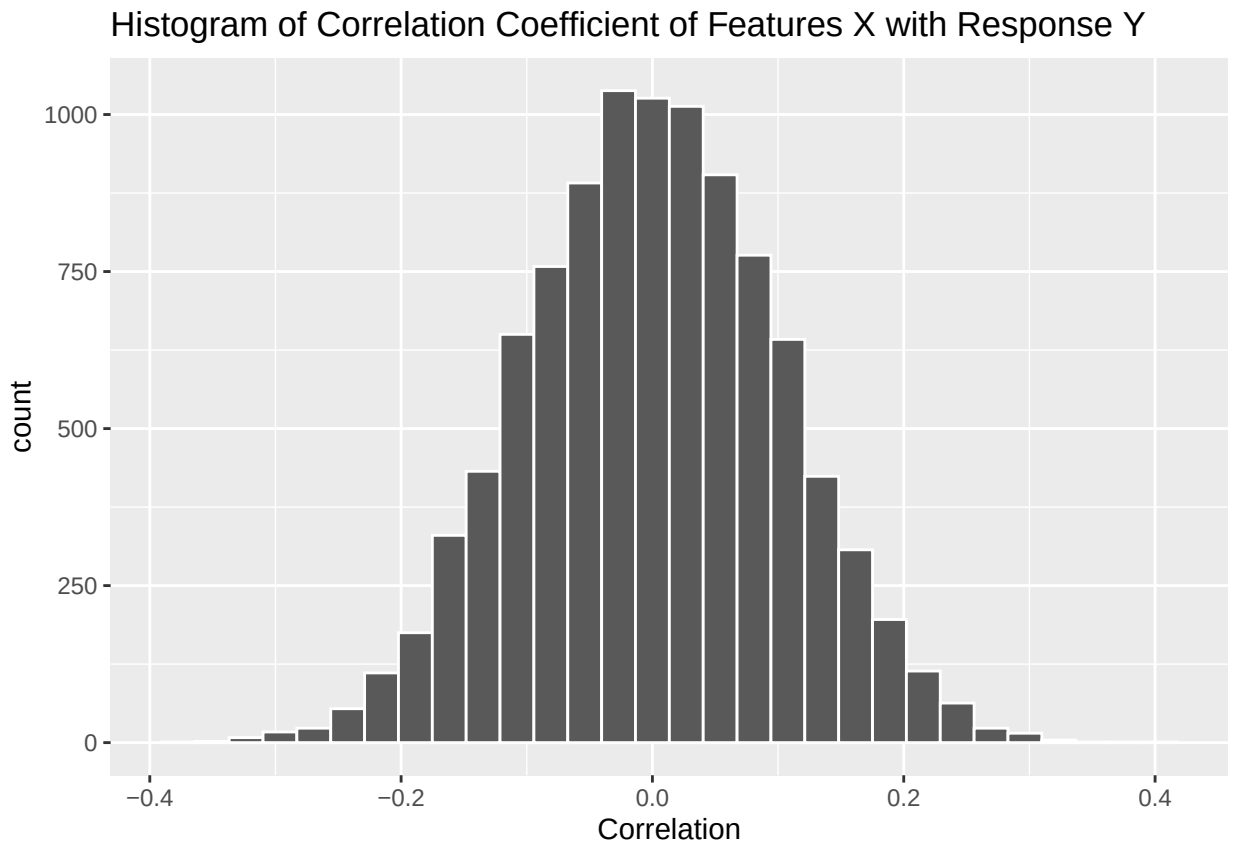
We also see that the theoretical test MSE is equal to the irreducible error and it is the lowest possible value of MSE we can achieve. Procedure in (d) will have a higher bias and variance, because we try to estimate f using $\hat{f}$ from a linear model.

2. In lecture during Week 5, we discussed "Option 1" and "Option 2": two possible ways to perform the validation set approach for a modeling strategy where you identify the q features most correlated with the response, and then fit a least squares linear model to predict the response using just those q features. If you missed that lecture, then please familiarize yourself with the lecture notes (posted on Canvas) before you continue. Here, we are going to continue to work with the simulated data from the previous problem, in order to illustrate the problem with Option 1.

(a) Calculate the correlation between each feature and the response. Make a histogram of these correlations. What are the values of the 10 largest absolute correlations?

```
cor.data <- data.frame(Correlation = t(cor(y,x)))
ggplot(data = cor.data, aes(x=Correlation)) +
  geom_histogram(color="white") +
  ggtitle(label = "Histogram of Correlation Coefficient of Features X with Response Y")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Values of the largest 10 absolute correlations:

```
#Sorting columns by highest absolute correlation
sorted.data <- sort(abs(cor.data$Correlation),
                    decreasing = TRUE, index.return = TRUE)

sorted.data$x[1:10]
```

```
##  [1] 0.4123726 0.3774119 0.3691199 0.3460830 0.3457064 0.3397968 0.3329436
##  [8] 0.3325739 0.3316668 0.3308413
```

(b) Now try out "Option 1" with $q = 10$. What is the estimated test error?

```
# Subsetting x by indexes of columns with highest correlation
x.2b <- x[,sorted.data$ix[1:10]]

# Fitting model and reporting MSE
data.2b <- data.frame(X = x.2b, y)

model.2b <- lm(y ~ ., data = data.2b, subset = rows)
```

```
mean((y - predict(model.2b, data.2b))[-rows]^2)
```

```
## [1] 0.4562992
```

(c) Now try out "Option 2" with $q = 10$. What is the estimated test error?

```
# Calculating highest correlation by absolute value on training set
sorted.data <- sort(abs(cor(x[rows,], y[rows])),
                    decreasing = TRUE, index.return = TRUE)

# Selecting top 10 features by absolute correlation
train <- train[, c(sorted.data$ix[1:10], 10001)]
val <- val[, c(sorted.data$ix[1:10], 10001)]

model.2c <- lm(y ~ ., data = train)
preds <- predict(model.2c, newdata = val)

mean((val[,"y"] - preds)^2)
```

```
## [1] 1.577791
```

(d) Comment on your results in (b) and (c). How does this relate to the discussion of Option 1 versus Option 2 from lecture? Explain how you can see that Option 1 gave you a useless (i.e. misleading, inaccurate, wrong) estimate of the test error.

We see that (b) gave us a much smaller test error than (c).

In the lecture we discussed why Option 1 is an incorrect method of feature selection because we first look at the entire data and then split it into training and validation sets. The problem with this method is that we are "peeking" into the validation set and have prior information about what that data looks like. This leads to grossly underestimating the test MSE.

To get a reliable estimate of the test MSE, we must separate the data into training and validation sets before we perform any data exploration or feature selection.

Option 1 gave us an inaccurate estimate of the test error because we know that the MSE consists of the irreducible error, squared bias and variance. We found in 1. (b) that the irreducible error is equal to 1. That means, any model that we create will have MSE > 1 because it will include irreducible error along with some bias and variance. In Option 1 we find that the MSE < 1, which is not possible.

3. In this problem, you will analyze a (real, not simulated) dataset of your choice with a quantitative response Y , and $p \geq 50$ quantitative predictors.

In this problem, you may use the function in the `glmnet` package that performs cross-validation.

(a) Describe the data. Where did you get it from? What is the meaning of the response, and what are the meanings of the predictors?

The data contains information about 21263 superconductors and 81 features that were captured for them. This data was acquired from the UCI Machine Learning Repository. It contains two files - we use only train.csv for this problem.

The goal is to predict the critical temperature of a superconductor given these features. The feature set consists of various statistics such as mean, weighted mean and geometric mean applied to physical and chemical characteristics of the elements. Below we list the various features present in the dataset:

```
data <- read.csv("superconduct_train.csv")

knitr::kable(data.frame(Feature = colnames(data),
           Type = c(rep("Predictor, Numeric", 81), "Response, Numeric")))
```

Feature	Type
number_of_elements	Predictor, Numeric
mean_atomic_mass	Predictor, Numeric
wtd_mean_atomic_mass	Predictor, Numeric
gmean_atomic_mass	Predictor, Numeric
wtd_gmean_atomic_mass	Predictor, Numeric
entropy_atomic_mass	Predictor, Numeric
wtd_entropy_atomic_mass	Predictor, Numeric
range_atomic_mass	Predictor, Numeric
wtd_range_atomic_mass	Predictor, Numeric
std_atomic_mass	Predictor, Numeric
wtd_std_atomic_mass	Predictor, Numeric
mean_fie	Predictor, Numeric
wtd_mean_fie	Predictor, Numeric
gmean_fie	Predictor, Numeric
wtd_gmean_fie	Predictor, Numeric
entropy_fie	Predictor, Numeric
wtd_entropy_fie	Predictor, Numeric
range_fie	Predictor, Numeric
wtd_range_fie	Predictor, Numeric
std_fie	Predictor, Numeric
wtd_std_fie	Predictor, Numeric
mean_atomic_radius	Predictor, Numeric
wtd_mean_atomic_radius	Predictor, Numeric
gmean_atomic_radius	Predictor, Numeric
wtd_gmean_atomic_radius	Predictor, Numeric
entropy_atomic_radius	Predictor, Numeric
wtd_entropy_atomic_radius	Predictor, Numeric
range_atomic_radius	Predictor, Numeric
wtd_range_atomic_radius	Predictor, Numeric
std_atomic_radius	Predictor, Numeric
wtd_std_atomic_radius	Predictor, Numeric
mean_Density	Predictor, Numeric
wtd_mean_Density	Predictor, Numeric
gmean_Density	Predictor, Numeric
wtd_gmean_Density	Predictor, Numeric
entropy_Density	Predictor, Numeric
wtd_entropy_Density	Predictor, Numeric
range_Density	Predictor, Numeric
wtd_range_Density	Predictor, Numeric
std_Density	Predictor, Numeric
wtd_std_Density	Predictor, Numeric
mean_ElectronAffinity	Predictor, Numeric
wtd_mean_ElectronAffinity	Predictor, Numeric
gmean_ElectronAffinity	Predictor, Numeric

Feature	Type
wtd_gmean_ElectronAffinity	Predictor, Numeric
entropy_ElectronAffinity	Predictor, Numeric
wtd_entropy_ElectronAffinity	Predictor, Numeric
range_ElectronAffinity	Predictor, Numeric
wtd_range_ElectronAffinity	Predictor, Numeric
std_ElectronAffinity	Predictor, Numeric
wtd_std_ElectronAffinity	Predictor, Numeric
mean_FusionHeat	Predictor, Numeric
wtd_mean_FusionHeat	Predictor, Numeric
gmean_FusionHeat	Predictor, Numeric
wtd_gmean_FusionHeat	Predictor, Numeric
entropy_FusionHeat	Predictor, Numeric
wtd_entropy_FusionHeat	Predictor, Numeric
range_FusionHeat	Predictor, Numeric
wtd_range_FusionHeat	Predictor, Numeric
std_FusionHeat	Predictor, Numeric
wtd_std_FusionHeat	Predictor, Numeric
mean_ThermalConductivity	Predictor, Numeric
wtd_mean_ThermalConductivity	Predictor, Numeric
gmean_ThermalConductivity	Predictor, Numeric
wtd_gmean_ThermalConductivity	Predictor, Numeric
entropy_ThermalConductivity	Predictor, Numeric
wtd_entropy_ThermalConductivity	Predictor, Numeric
range_ThermalConductivity	Predictor, Numeric
wtd_range_ThermalConductivity	Predictor, Numeric
std_ThermalConductivity	Predictor, Numeric
wtd_std_ThermalConductivity	Predictor, Numeric
mean_Valence	Predictor, Numeric
wtd_mean_Valence	Predictor, Numeric
gmean_Valence	Predictor, Numeric
wtd_gmean_Valence	Predictor, Numeric
entropy_Valence	Predictor, Numeric
wtd_entropy_Valence	Predictor, Numeric
range_Valence	Predictor, Numeric
wtd_range_Valence	Predictor, Numeric
std_Valence	Predictor, Numeric
wtd_std_Valence	Predictor, Numeric
critical_temp	Response, Numeric

(b) Fit a least squares linear model to the data, and provide an estimate of the test error. (Explain how you got this estimate.)

```
set.seed(42)

train <- sample(nrow(data), 0.5*nrow(data), replace = FALSE)
model.3b <- lm(critical_temp ~ ., data = data, subset = train)

mean((data[,"critical_temp"] - predict(model.3b, data))[-train]^2)
```

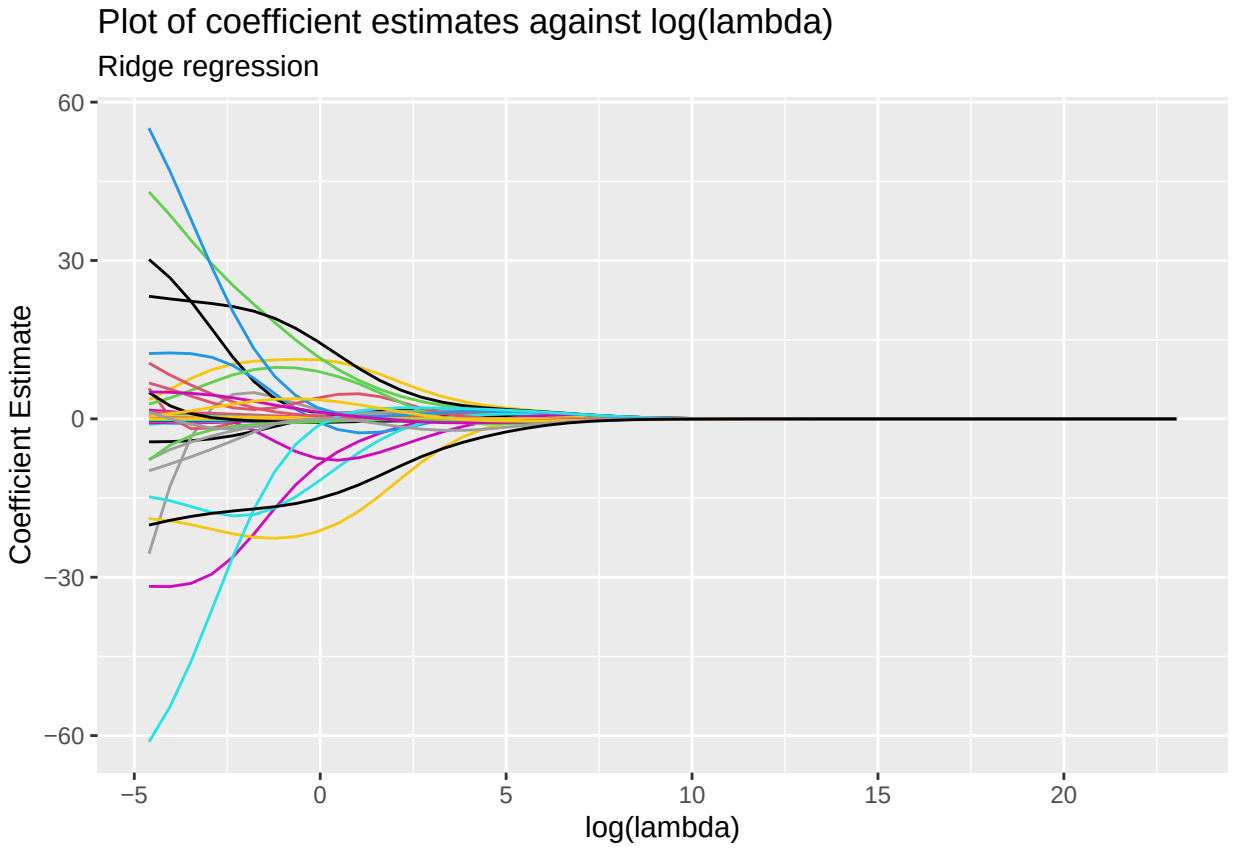
```
## [1] 316.0399
```

The test error is obtained by using a validation set. We fit the model on only the training data and then test

this model's fit on the validation set.

(c) Fit a ridge regression model to the data, with a range of values of the tuning parameter λ. Make a plot like the left-hand panel of Figure 6.4 in the textbook.

```
x <- model.matrix(critical_temp ~ ., data = data)[,-1]
y <- data[,"critical_temp"]

grid <- 10^seq(10, -2, length = 50)


plot.regularization <- function(alpha=0){

  model <- glmnet(x[train,], y[train], lambda = grid, alpha = alpha, thresh = 1e-12)

  betas <- model$beta

  plots <- ggplot()
  for(i in 1:nrow(betas)){
    df <- data.frame(coef = betas[i,], lambda = grid)
    plots <- plots + geom_line(data = df, aes(x=log(lambda), y=coef), color = i)
  }
  plots <- plots +
    labs(title = "Plot of coefficient estimates against log(lambda)",
         y = "Coefficient Estimate")
  return(plots)
}

(ridge.plot <- plot.regularization() + labs(subtitle = "Ridge regression"))
```



(d) What value of λ in the ridge regression model provides the smallest estimated test error? Report this estimate of test error. (Also, explain how you estimated test error.)

```
set.seed(42)

cv.ridge <- cv.glmnet(x[train,], y[train], alpha=0, lambda = grid)
best.lambda.ridge <- cv.ridge$lambda.min
```

We find using cross validation on the training data that the best value of λ = 0.01

```
best.ridge <- predict(cv.ridge, s = best.lambda.ridge, newx = x[-train,])
mean((best.ridge - y[-train])^2)
```

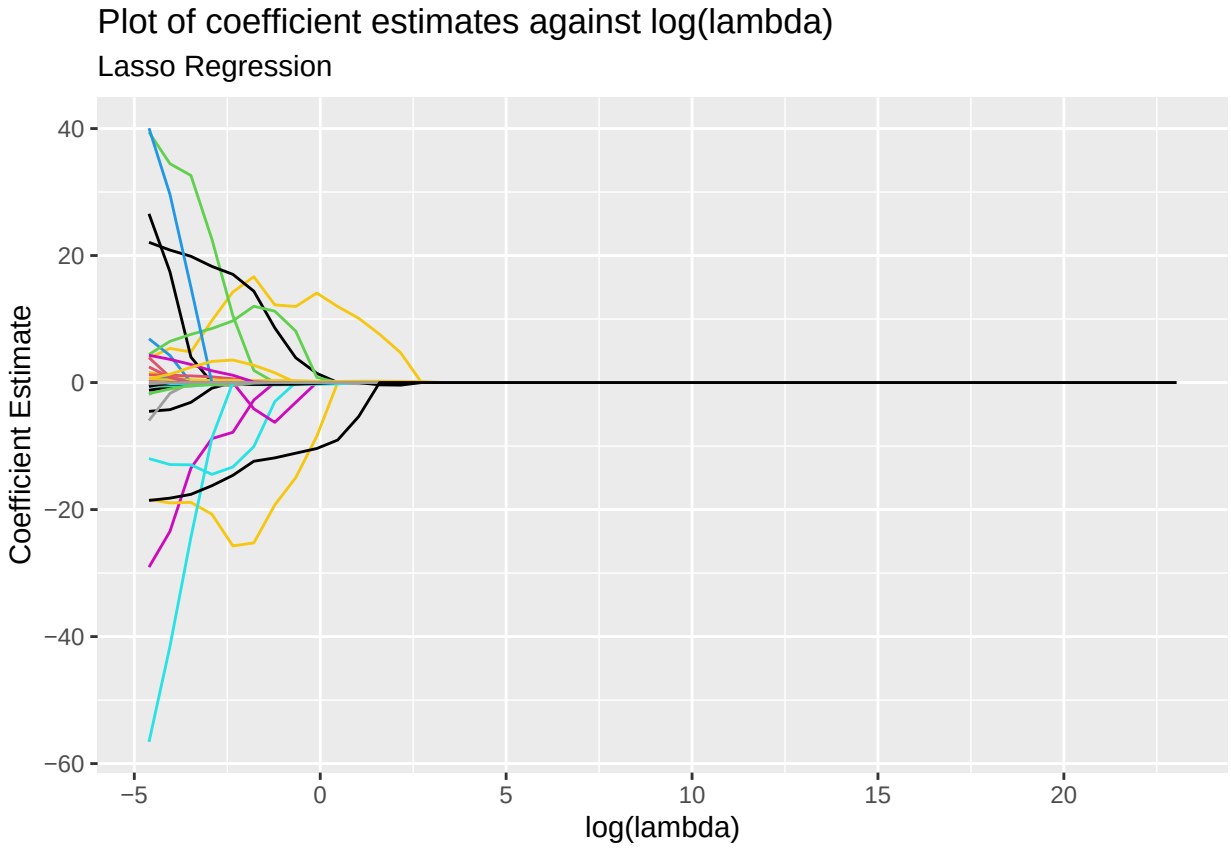
```
## [1] 318.0136
```

The estimated test error is stated above. We get this test error by first, identifying the best value of λ by running cross validation on the training data and then calculating MSE on the held out data that was not used in the cross validation process.

The final ridge regression model that we deploy into the real world should be re-trained on all the data

(e) Repeat (c), but for a lasso model.

```
(lasso.plot <- plot.regularization(alpha=1) + labs(subtitle = "Lasso Regression"))
```



(f) Repeat (d), but for a lasso model. Which features are included in this lasso model?

```
set.seed(42)

cv.lasso <- cv.glmnet(x[train,], y[train], alpha=1, lambda = grid)
best.lambda.lasso <- cv.lasso$lambda.min
```

We find using cross validation on the training data that the best value of λ = 0.01

```
best.lasso <- predict(cv.lasso, s = best.lambda.lasso, newx = x[-train,])
mean((best.lasso - y[-train])^2)
```

```
## [1] 320.4264
```

The estimated test error is stated above. We get this test error by first, identifying the best value of λ by running cross validation on the training data and then calculating MSE on the held out data that was not used in the cross validation process.

The features included in the model are:

```
refit.lasso.model <- glmnet(x, y, lambda = grid, alpha = 1)
lasso.coef <- predict(refit.lasso.model, s = best.lambda.lasso,
                      type = "coefficients")
lasso.df <- data.frame(Variable = names(lasso.coef[1:82,]),
                       Coefficient = as.numeric(lasso.coef))

# Variables included in model
```

```
knitr::kable(lasso.df |> filter(Coefficient != 0))
```

Variable	Coefficient
(Intercept)	-18.9027973
number_of_elements	-3.9419328
mean_atomic_mass	0.3006913
wtd_mean_atomic_mass	-0.2123760
gmean_atomic_mass	-0.0000638
entropy_atomic_mass	-33.9420131
wtd_entropy_atomic_mass	9.5613407
range_atomic_mass	0.1850382
wtd_range_atomic_mass	0.0052798
std_atomic_mass	-0.2030454
wtd_std_atomic_mass	-0.2330932
mean_fie	0.0041770
wtd_mean_fie	0.0171060
wtd_entropy_fie	31.6758310
range_fie	0.0636609
wtd_range_fie	0.0177498
std_fie	-0.1535425
wtd_std_fie	-0.0269597
mean_atomic_radius	0.6658784
wtd_mean_atomic_radius	0.7257185
gmean_atomic_radius	-0.9112307
wtd_gmean_atomic_radius	-0.4040733
wtd_entropy_atomic_radius	25.1565940
range_atomic_radius	0.2082621
wtd_range_atomic_radius	-0.0938541
std_atomic_radius	-0.8482341
wtd_std_atomic_radius	0.3428279
mean_Density	-0.0035001
wtd_mean_Density	-0.0002375
gmean_Density	0.0001469
wtd_gmean_Density	0.0022306
entropy_Density	12.3246151
wtd_entropy_Density	-17.1588558
range_Density	-0.0011535
wtd_range_Density	0.0000349
std_Density	0.0039848
wtd_std_Density	-0.0009605
mean_ElectronAffinity	-0.0836372
wtd_mean_ElectronAffinity	0.5328685
gmean_ElectronAffinity	0.1580488
wtd_gmean_ElectronAffinity	-0.5810037
entropy_ElectronAffinity	1.2353083
wtd_entropy_ElectronAffinity	-19.9769929
range_ElectronAffinity	-0.3605139
wtd_range_ElectronAffinity	-0.1599218
std_ElectronAffinity	1.1891791
wtd_std_ElectronAffinity	-0.5177181
mean_FusionHeat	0.3021391
wtd_mean_FusionHeat	-0.3775906
gmean_FusionHeat	-0.2018553

Variable	Coefficient
wtd_gmean_FusionHeat	0.1661081
entropy_FusionHeat	-9.4040362
wtd_entropy_FusionHeat	22.0252511
range_FusionHeat	-0.3034763
wtd_range_FusionHeat	0.4256027
std_FusionHeat	0.2910006
wtd_std_FusionHeat	-0.1906372
mean_ThermalConductivity	-0.0109696
wtd_mean_ThermalConductivity	0.4364737
gmean_ThermalConductivity	-0.0940850
wtd_gmean_ThermalConductivity	-0.2586973
entropy_ThermalConductivity	3.9773253
wtd_entropy_ThermalConductivity	6.3386956
range_ThermalConductivity	-0.0501505
wtd_range_ThermalConductivity	-0.1823377
std_ThermalConductivity	0.1012311
wtd_std_ThermalConductivity	0.0658615
wtd_mean_Valence	-0.8827215
gmean_Valence	2.6534433
wtd_gmean_Valence	-2.2075350
entropy_Valence	34.0993910
wtd_entropy_Valence	-47.0639324
range_Valence	4.1277136
wtd_range_Valence	1.2757335
wtd_std_Valence	-18.5343455

```
# Variables excluded from model
knitr::kable(lasso.df |> filter(Coefficient == 0))
```

Variable	Coefficient
wtd_gmean_atomic_mass	0
gmean_fie	0
wtd_gmean_fie	0
entropy_fie	0
entropy_atomic_radius	0
mean_Valence	0
std_Valence	0

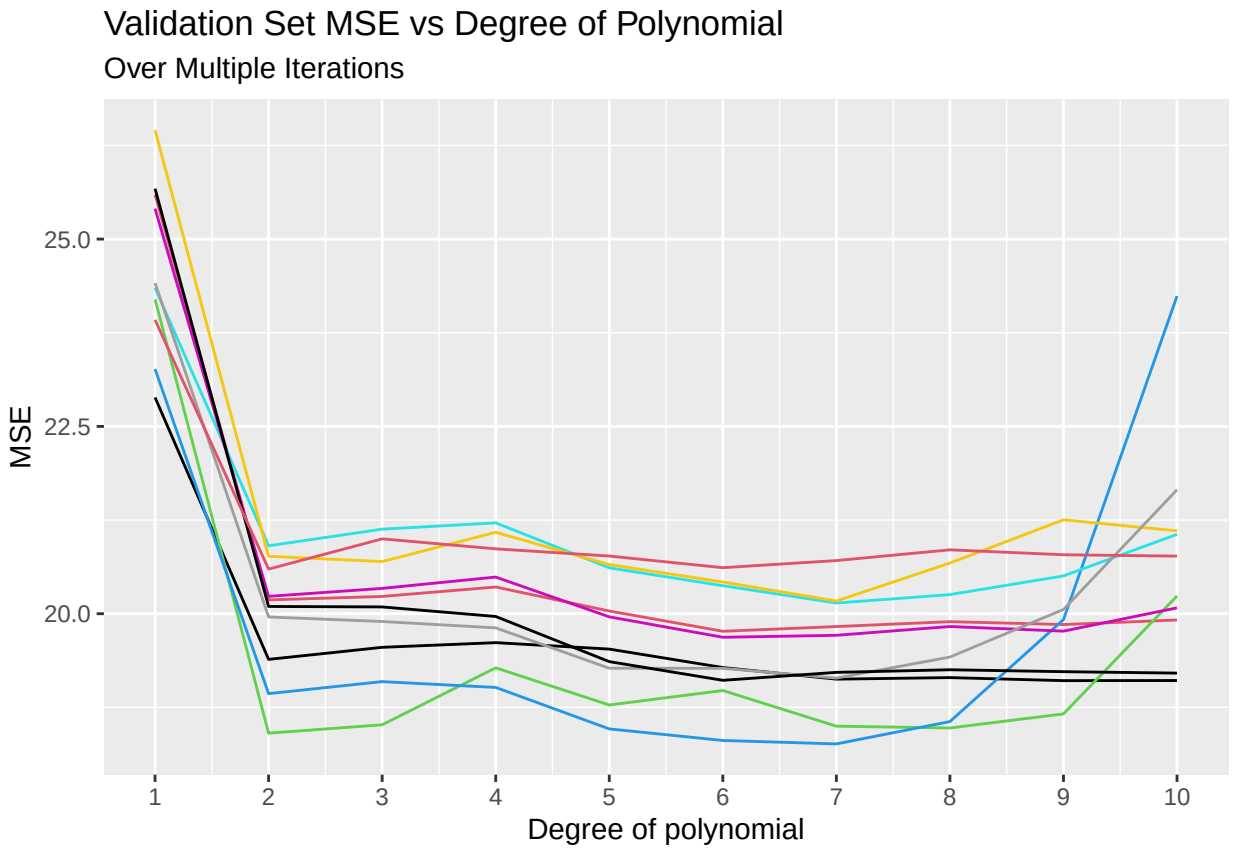
4.Consider using the `Auto` data set to predict `mpg` using polynomial functions of `horsepower` in a least squares linear regression.

(a) Perform the validation set approach, and produce a plot like the one in the right-hand panel of Figure 5.2 of the textbook. Your answer won't look *exactly* the same as the results in Figure 5.2, since you'll be starting with a different random seed. Discuss your findings. What degree polynomial is best, and why?

```r
set.seed(42)
attach(Auto)

# Defining function to repeat cross-validation

perform.cv <- function(all = FALSE){
  # all = TRUE allows us to train model on entire training set
  if(all){
   train <- sample(nrow(Auto), nrow(Auto), replace = FALSE)
  }
  else{
    train <- sample(nrow(Auto), 0.5*(nrow(Auto)))
  }
  D <- 1:10
  mse <- rep(0,length(D))
  for(d in D){
    model <- lm(mpg ~ poly(horsepower, d), subset = train)
    if(all){
      mse[d] <- mean((mpg - predict(model, Auto))^2)
    }
    else{
      mse[d] <- mean((mpg - predict(model, Auto))[-train]^2)
    }
  }
  return(mse)
}
```

Above, we define a function to perform cross validation over the auto dataset for each degree of polynomial. We will now call this repeatedly.

```r
mse <- matrix(rep(0, 10*10), nrow = 10)

plots <- ggplot()
for(i in 1:10){
  mse[i,] <-  perform.cv()
  df <- data.frame(poly = 1:10, MSE = mse[i,])
  plots <- plots + geom_line(data = df, aes(x=poly, y=MSE), color=i)
}

plots +
  scale_x_continuous(breaks = c(1:10)) +
  labs(x = "Degree of polynomial",
       title = "Validation Set MSE vs Degree of Polynomial",
       subtitle = "Over Multiple Iterations ")
```

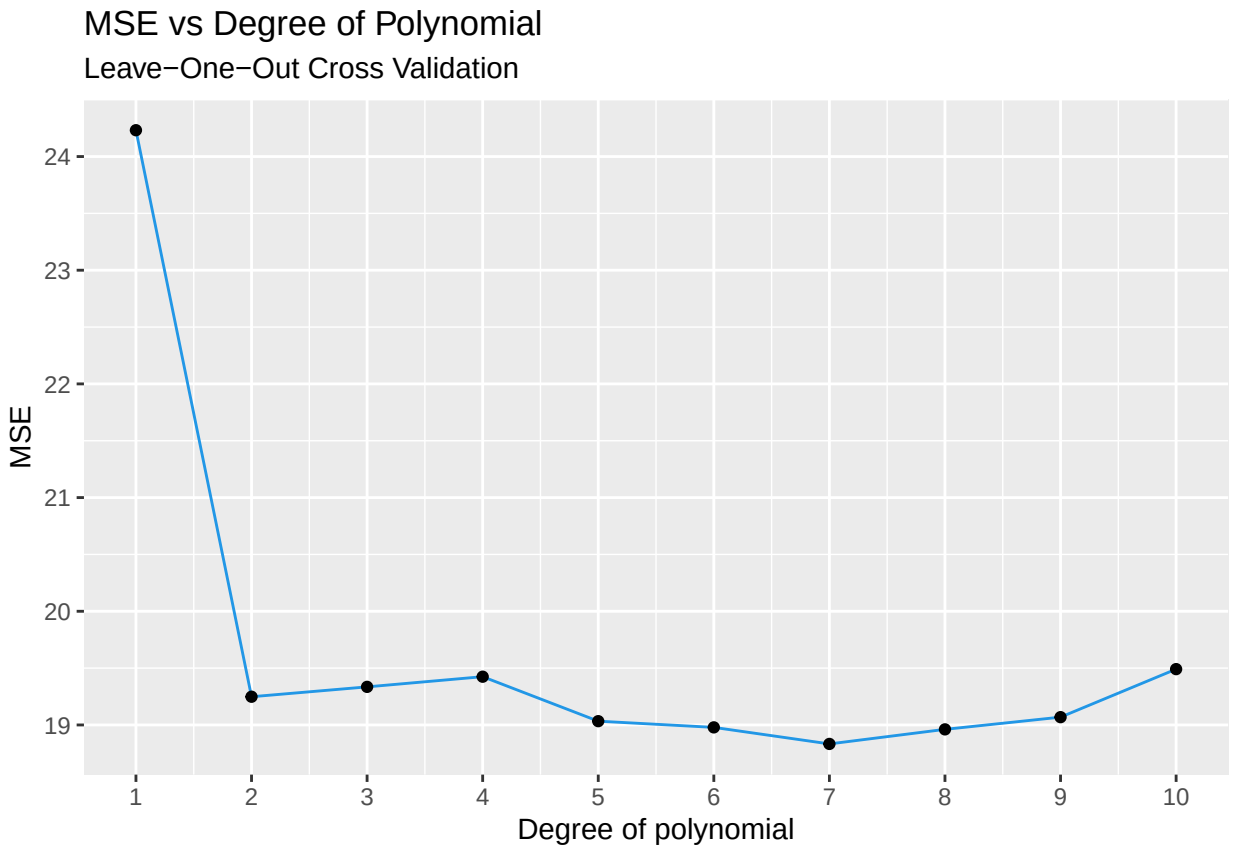


We find that adding higher order polynomials of horsepower is generally better. There is a lot of variance in MSE between each run due the the random split used to generate training data but the trends of each iteration are mostly in agreement - degree 2 is considerably better than degree 1. We see that for very high order polynomials (> 7) that MSE starts to increase in some runs.

A degree 2 polynomial appears to be the best choice because it produces a huge reduction in MSE compared to a degree 1 polynomial and has MSE similar to other higher order polynomials while being a much simpler model (thereby reducing variance).

(b) Perform leave-one-out cross-validation, and produce a plot like the one in the left-hand panel of Figure 5.4 of the textbook. Discuss your findings. What degree polynomial is best, and why?

```
loocv.mse <- rep(0, 10)

for(i in 1:10){
  glm.fit <- glm(mpg ~ poly(horsepower, i), data = Auto)
  loocv.mse[i] <- cv.glm(Auto, glm.fit)$delta[1]
}

ggplot(data = data.frame(d = 1:10, loocv.mse), aes(x=d, y=loocv.mse)) +
  geom_line(color=20) + geom_point()+
  scale_x_continuous(breaks = c(1:10)) +
  labs(x = "Degree of polynomial", y="MSE",
       title = "MSE vs Degree of Polynomial",
       subtitle = "Leave-One-Out Cross Validation")
```



Similar to (a) we see that adding higher order polynomials greatly reduces the MSE. Polynomial of degree 7 has the lowest MSE but it still seems like degree 2 might be the best option due to being a much simpler model and still having MSE comparable to degree 7.

(c) Perform 10-fold cross-validation, and produce a plot like the one in the right-hand panel of Figure 5.4 of the textbook. Discuss your findings. What degree polynomial is best, and why?

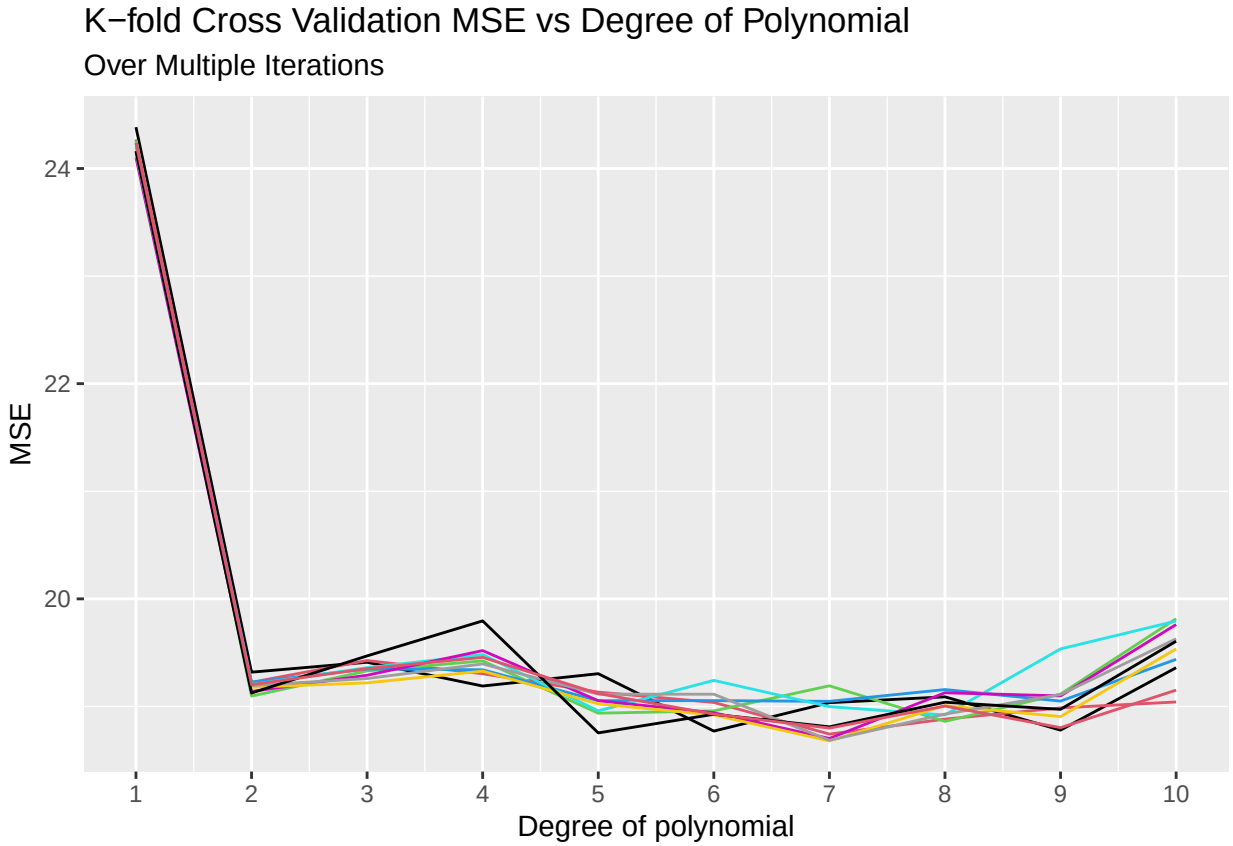
Defining a function to perform k-fold cross validation.

```
set.seed(42)

perform.kfold.cv <- function(k=10) {
 kfold.cv <- rep(0,10)
 for(i in 1:10){
   glm.fit <- glm(mpg ~ poly(horsepower, i), data = Auto)
   kfold.cv[i] <- cv.glm(Auto, glm.fit, K = k)$delta[1]
 }
 return(kfold.cv)
}
```

Now, we call this function repeatedly.

```
mse <- matrix(rep(0, 10*10), nrow = 10)

plots <- ggplot()
for(i in 1:10){
```

```
  mse[i,] <-  perform.kfold.cv()
  df <- data.frame(poly = 1:10, MSE = mse[i,])
  plots <- plots + geom_line(data = df, aes(x=poly, y=MSE), color=i)
}

plots +
  scale_x_continuous(breaks = c(1:10)) +
  labs(x = "Degree of polynomial",
       title = "K-fold Cross Validation MSE vs Degree of Polynomial",
       subtitle = "Over Multiple Iterations ")
```



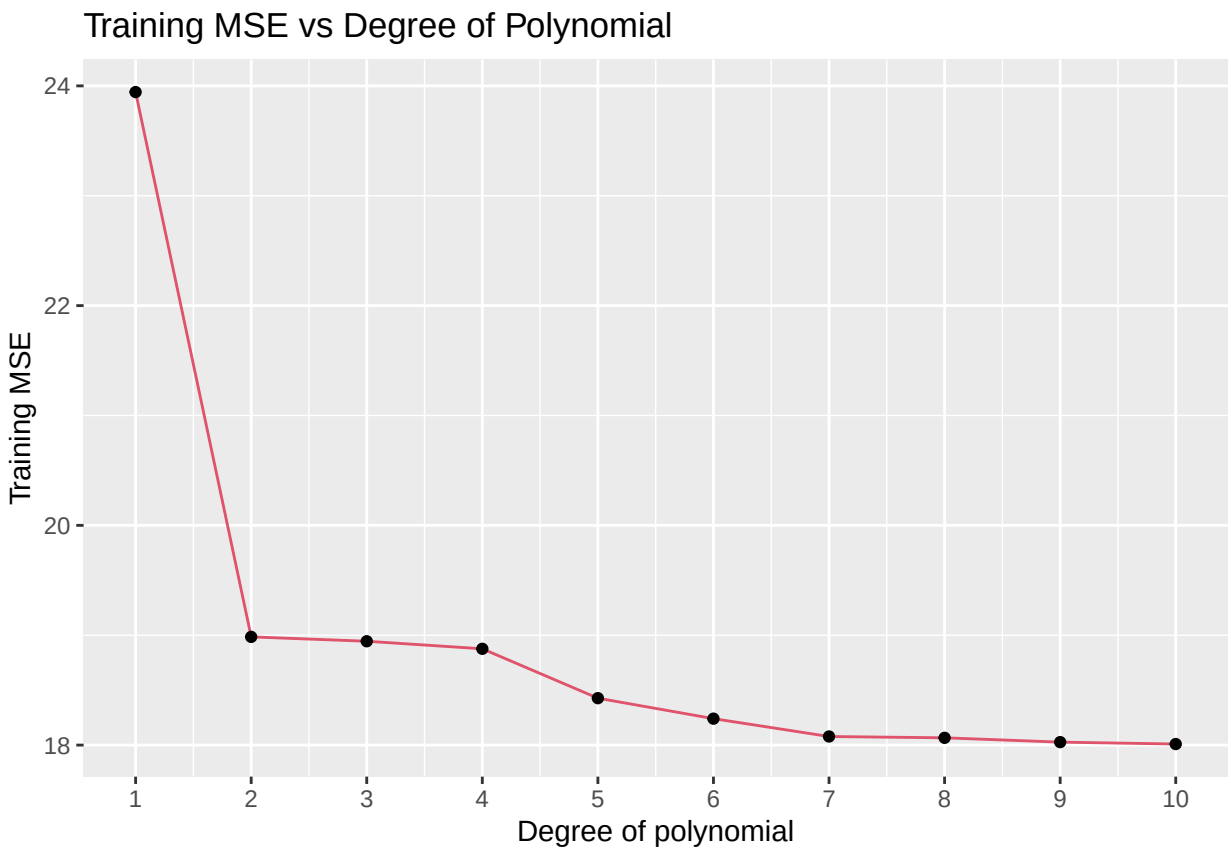
We find that the variance in MSE over runs is much lower in k-fold cross validation compared to the validation set approach over multiple runs. In most runs it appears that a degree 7 polynomial has the lowest MSE, however there is more variability in degree 7's MSE compared to degree 2 over multiple runs.

Given the consistency in results and model simplicity, a degree 2 polynomial is the best option to capture the relation between horsepower and mpg.

(d) Fit a least squares linear model to predict `mpg` using polynomials of degrees from 1 to 10, using all available observations. Make a plot showing "Degree of Polynomial" on the *x*-axis, and "Training Set Mean Squared Error" on the *y*-axis. Discuss your findings.

```
train.mse <- perform.cv(all = TRUE)

ggplot(data = data.frame(d=1:10, train.mse), aes(x=d, y=train.mse))+
```

```
  geom_line(color=10)+geom_point()+
  scale_x_continuous(breaks = c(1:10))+
  labs(x = "Degree of polynomial", y="Training MSE",
       title = "Training MSE vs Degree of Polynomial")
```



We find that the training MSE consistently reduces as we increase the degree of polynomial

(e) Fit a least squares linear model to predict `mpg` using a degree-10 polynomial, using all available observations. Using the `summary` command in `R`, examine the output. Comment on the output, and discuss how this relates to your findings in (a)–(d).

```
model.4e <- lm(mpg ~ poly(horsepower, 10), data = Auto)

summary(model.4e)
```

```
##
## Call:
## lm(formula = mpg ~ poly(horsepower, 10), data = Auto)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.7081  -2.5904  -0.1922   2.2859  14.8338
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)              23.4459     0.2174 107.840   <2e-16 ***
## poly(horsepower, 10)1  -120.1377     4.3046 -27.909   <2e-16 ***
## poly(horsepower, 10)2    44.0895     4.3046  10.242   <2e-16 ***
## poly(horsepower, 10)3    -3.9488     4.3046  -0.917   0.3595
## poly(horsepower, 10)4    -5.1878     4.3046  -1.205   0.2289
## poly(horsepower, 10)5    13.2722     4.3046   3.083   0.0022 **
## poly(horsepower, 10)6    -8.5462     4.3046  -1.985   0.0478 *
## poly(horsepower, 10)7     7.9806     4.3046   1.854   0.0645 .
## poly(horsepower, 10)8     2.1727     4.3046   0.505   0.6140
## poly(horsepower, 10)9    -3.9182     4.3046  -0.910   0.3633
## poly(horsepower, 10)10   -2.6146     4.3046  -0.607   0.5440
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.305 on 381 degrees of freedom
## Multiple R-squared:  0.7036, Adjusted R-squared:  0.6958
## F-statistic: 90.45 on 10 and 381 DF,  p-value: < 2.2e-16
```

Based on the above summary of the model the coefficient of the degree 10 polynomial is not statistically significant when added to the model, i.e. it offers no improvement. This relates to the findings in (a)-(c) where we found that the MSE increases for degree 10 in all the cross validation approaches. The reason the training MSE decreases in (d) is because adding higher order features will always reduce the residual sum of squares but causes the model to overfit - the training MSE will be low, however MSE on a test set with unseen observations would be high.

5. *Extra Credit!* Let's consider doing least squares and ridge regression under a very simple setting, in which $p = 1$, and $\sum_{i=1}^{n} y_i = \sum_{i=1}^{n} x_i = 0$. We consider regression without an intercept. (It's usually a bad idea to do regression without an intercept, but if our feature and response each have mean zero, then it is okay to do this!)

Hint: For this problem, you might want to brush up on some basic properties of means and variances! For instance, if $Cov(Z, W) = 0$, then $Var(Z + W) = Var(Z) + Var(W)$. And if a is a constant, then $Var(aW) = a^2 Var(W)$, and $Var(a + W) = Var(W)$.

(a) The least squares solution is the value of $\beta \in \mathbb{R}$ that minimizes $\sum_{i=1}^{n}(y_i - \beta x_i)^2$. Write out an analytical (closed-form) expression for this least squares solution. Your answer should be a function of $x_1, ..., x_n$ and $y_1, ..., y_n$. *Hint: Calculus!!*

The least squares solution will require taking the partial derivative of $\sum_{i=1}^{n}(y_i - \beta x_i)^2$ with respect to β.

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

$$0 = \sum_{i=1}^{n} x_i y_i - \beta x_i^2 \tag{3}$$

$$\hat{\beta} = \frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2}$$

(b) For a given value of λ, the ridge regression solution minimizes $\sum_{i=1}^{n}(y_i - \beta x_i)^2 + \lambda\beta^2$. Write out an analytical (closed-form) expression for the ridge regression solution, in terms of $x_1, ..., x_n$ and $y_1, ..., y_n$ and λ.

For the ridge solution, we perform the same operation as (a) but this time on the equation $\sum_{i=1}^{n}(y_i - \beta x_i)^2 + \lambda\beta^2$.

$$\frac{\partial}{\partial\beta}\sum_{i=1}^{n}(y_i-\beta x_i)^2 + \lambda\beta^2 = -2\sum_{i=1}^{n}(y_i-\beta x_i)x_i + 2\lambda\beta$$

$$\sum_{i=1}^{n} x_i y_i = \lambda\beta + \sum_{i=1}^{n}\beta x_i^2 \tag{4}$$

$$\hat{\beta_R} = \frac{\sum_{i=1}^{n} x_i y_i}{(\lambda + \sum_{i=1}^{n} x_i^2)}$$

(c) Suppose that the true data-generating model is $Y = 3X + \varepsilon$, where ε has mean zero, and X is fixed (non-random). What is the expectation of the least squares estimator from (a)? Is it biased or unbiased?

$$\begin{aligned} E[\hat{\beta}] &= E\left[\frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2}\right] \\ &= E\left[\frac{\sum_{i=1}^{n} x_i(3x_i+\varepsilon_i)}{\sum_{i=1}^{n} x_i^2}\right] \\ &= E\left[\frac{3\sum_{i=1}^{n} x_i^2 + x_i\varepsilon_i}{\sum_{i=1}^{n} x_i^2}\right] \\ &= 3\frac{\sum_{i=1}^{n} x_i^2}{\sum_{i=1}^{n} x_i^2} + \frac{\sum_{i=1}^{n} x_i E[\varepsilon_i]}{\sum_{i=1}^{n} x_i^2} \\ E[\hat{\beta}] &= 3 \end{aligned} \tag{5}$$

We get the estimated value of β to be 3. This is because we know $E[\varepsilon_i] = 0$. This turns out to be an unbiased estimator because $E[\hat{\beta}] = \beta$.

(d) Suppose again that the true data-generating model is $Y = 3X + \varepsilon$, where ε has mean zero, and X is fixed (non-random). What is the expectation of the ridge regression estimator from (b)? Is it biased or unbiased? Explain how the bias changes as a function of λ.

$$\begin{aligned} E[\widehat{\beta}_R] &= E\left[\frac{\sum_{i=1}^n x_i y_i}{\lambda + \sum_{i=1}^n x_i^2}\right] \\ &= E\left[\frac{3\sum_{i=1}^n x_i^2 + x_i\varepsilon_i}{\lambda + \sum_{i=1}^n x_i^2}\right] \\ &= 3\frac{\sum_{i=1}^n x_i^2}{\lambda + \sum_{i=1}^n x_i^2} + \frac{\sum_{i=1}^n x_i E[\varepsilon_i]}{\lambda + \sum_{i=1}^n x_i^2} \\ E[\widehat{\beta}_R] &= 3\frac{\sum_{i=1}^n x_i^2}{\lambda + \sum_{i=1}^n x_i^2} (\because E[\varepsilon_i] = 0) \end{aligned} \tag{6}$$

We see that the ridge regression coefficient estimate is a biased estimator of β. As $\lambda \to \infty, E[\widehat{\beta}_R] = 0$ adding more bias to the estimate, and when $\lambda = 0, E[\widehat{\beta}_R] = 3$, i.e. when $\lambda = 0$ the ridge estimator is equivalent to the least squares estimator.

(e) Suppose that the true data-generating model is $Y = 3X + \varepsilon$, where ε has mean zero and variance σ^2, and X is fixed (non-random), and also $Cov(\varepsilon_i, \varepsilon_{i'}) = 0$ for all $i \neq i'$. What is the variance of the least squares estimator from (a)?

$$
\begin{aligned}
Var(\hat{\beta}) &= Var\left(\frac{\sum_{i=1}^n x_i y_i}{\sum_{i=1}^n x_i^2}\right) \\
&= Var\left(\frac{3\sum_{i=1}^n x_i^2 + x_i\varepsilon_i}{\sum_{i=1}^n x_i^2}\right) \\
&= Var\left(3\frac{\sum_{i=1}^n x_i^2}{\sum_{i=1}^n x_i^2} + \frac{\sum_{i=1}^n x_i\varepsilon_i}{\sum_{i=1}^n x_i^2}\right) \\
&= Var(3) + Var\left(\frac{\sum_{i=1}^n x_i\varepsilon_i}{\sum_{i=1}^n x_i^2}\right) \\
&= 0 + \left(\frac{1}{\sum_{i=1}^n x_i^2}\right)^2 Var(x_1\varepsilon_1 + ... + x_n\varepsilon_n) \qquad (7) \\
&= \left(\frac{1}{\sum_{i=1}^n x_i^2}\right)^2 (x_1^2 Var(\varepsilon_1) + ... + x_n^2 Var(\varepsilon_n)) \\
&= \left(\frac{1}{\sum_{i=1}^n x_i^2}\right)^2 (x_1^2\sigma^2 + ... + x_n^2\sigma^2)(\because Cov(\varepsilon_i, \varepsilon_{i'}) = 0) \\
&= \left(\frac{1}{\sum_{i=1}^n x_i^2}\right)^2 n\sigma^2 \sum_{i=1}^n x_i^2 \\
Var(\hat{\beta}) &= \frac{n\sigma^2}{\sum_{i=1}^n x_i^2}
\end{aligned}
$$

(f) Suppose that the true data-generating model is $Y = 3X + \varepsilon$, where ε has mean zero and variance σ^2, and X is fixed (non-random), and also $Cov(\varepsilon_i, \varepsilon_{i'}) = 0$ for all $i \neq i'$. What is the variance of the ridge estimator from (b)? How does the variance change as a function of λ?

$$
\begin{aligned}
Var(\hat{\beta}_R) &= Var\left(\frac{\sum_{i=1}^n x_i y_i}{\lambda + \sum_{i=1}^n x_i^2}\right) \\
&= Var\left(\frac{3\sum_{i=1}^n x_i^2 + x_i\varepsilon_i}{\lambda + \sum_{i=1}^n x_i^2}\right) \\
&= Var\left(3\frac{\sum_{i=1}^n x_i^2}{\lambda + \sum_{i=1}^n x_i^2} + \frac{\sum_{i=1}^n x_i\varepsilon_i}{\lambda + \sum_{i=1}^n x_i^2}\right) \\
&= Var\left(3\frac{\sum_{i=1}^n x_i^2}{\lambda + \sum_{i=1}^n x_i^2}\right) + Var\left(\frac{\sum_{i=1}^n x_i\varepsilon_i}{\lambda + \sum_{i=1}^n x_i^2}\right) \\
&= 0 + Var\left(\frac{\sum_{i=1}^n x_i\varepsilon_i}{\lambda + \sum_{i=1}^n x_i^2}\right) (\because Var(a) = 0) \\
&= \left(\frac{1}{\lambda + \sum_{i=1}^n x_i^2}\right)^2 Var(x_1\varepsilon_1 + ... + x_n\varepsilon_n) \\
&= \left(\frac{1}{\lambda + \sum_{i=1}^n x_i^2}\right)^2 (x_1^2 Var(\varepsilon_1) + ... + x_n^2 Var(\varepsilon_n)) \\
&= \left(\frac{1}{\lambda + \sum_{i=1}^n x_i^2}\right)^2 (x_1^2\sigma^2 + ... + x_n^2\sigma^2)(\because Cov(\varepsilon_i, \varepsilon_{i'}) = 0) \\
Var(\hat{\beta}_R) &= \frac{\sum_{i=1}^n x_i^2}{(\lambda + \sum_{i=1}^n x_i^2)^2} n\sigma^2
\end{aligned}
\tag{8}
$$

We see that as we increase the value of λ the variance reduces because the quantity $(\lambda + \sum_{i=1}^n x_i^2)^2 > \sum_{i=1}^n x_i^2$.

(g) In light of your answers to parts (d) and (f), argue that λ in ridge regression allows us to control model complexity by trading off bias for variance.

We learned from (d) that

$$Bias_R = \beta - E[\hat{\beta}_R]$$

where $E[\hat{\beta}_R] = 3\frac{\sum_{i=1}^n x_i^2}{\lambda + \sum_{i=1}^n x_i^2}$. For any $\lambda > 0$, we see that the denominator is greater than the numerator making the value of $E[\hat{\beta}_R] < 3$. Therefore we will have a positive value for Bias when we include a non-zero lambda because we are subtracting a value smaller than 3 from 3 when $\lambda > 0$.

We also know that least squares regression gives an unbiased estimator of β i.e. $Bias_{LS} = 0$. Therefore we see that ridge regression introduces some bias compared to least squares regression.

Now, we know that the ridge regression variance is

$$Var(\hat{\beta_R}) = \frac{\sum_{i=1}^{n} x_i^2}{(\lambda + \sum_{i=1}^{n} x_i^2)^2} n\sigma^2$$

And the variance for the least squares estimator is,

$$Var(\hat{\beta}) = \frac{n\sigma^2}{\sum_{i=1}^{n} x_i^2}$$

We need to show when $Var(\hat{\beta_R}) < Var(\hat{\beta})$ to establish that ridge regression trades off higher bias for decrease in variance.

Let $\sum_{i=1}^{n} x_i^2 = s$.

$$\begin{aligned}
Var(\hat{\beta_R}) &< Var(\hat{\beta}) \\
\frac{s}{(\lambda + s)^2} n\sigma^2 &< \frac{n\sigma^2}{s} \\
\frac{s}{(\lambda + s)^2} &< \frac{1}{s} \\
\frac{1}{(\lambda + s)^2} &< \frac{1}{s^2} \\
(\lambda + s)^2 &> s^2 \\
\lambda + s &> s \\
\implies \lambda &> 0
\end{aligned} \tag{9}$$

We see that the above inequality holds true when $\lambda > 0$, which it turns out is always true when we perform ridge regression. Hence we have shown that, ridge regression has a higher bias, but has lower variance than least squares regression. Controlling the value of λ allows us to control the bias and variance. Adding more bias means that we are reducing model flexibility and hence complexity of how the model represents data.