Stat 432 HW 02

Name: Ahmadreza Eslaminia, netID: ae15

Summer 2024

Include the R code for this HW.

```
knitr::opts_chunk$set(echo = TRUE)
library(ISLR2)
library(GGally)
```

There are some useful R chunk options that you may use (for this entire semester):

- echo - Display code in output document (default = TRUE)
- include - Include chunk in document after running (default = TRUE)
- message - display code messages in document (default = TRUE)
- results (default = 'markup')
  - 'asis' - passthrough results
  - 'hide' - do not display results
  - 'hold' - put all results below all code
- error - Display error messages in doc (TRUE) or stop render when errors occur (FALSE) (default = FALSE)

See R markdown cheat sheet for more information.

Question 1 (Linear Regression)

We have N observations of $(X_1, X_2, \ldots, X_p, Y)$.

Let us use the following notations:

- $\mathbf{X}$ is a the $N \times (p+1)$ matrix with each row as an input vector (with a 1 in the first position),
- $\mathbf{y}$ be the N-vector of outputs and
- $\boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{bmatrix}$.

Then we may write the multiple linear regression model as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \epsilon.$$

Show that

$$\widehat{\boldsymbol{\beta}} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}.$$

minimizes RSS.

Answer: for the linear regression model:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

where: - $\mathbf{X}$ is an $N \times (p+1)$ matrix of inputs. - $\mathbf{y}$ is an N-vector of outputs. - $\boldsymbol{\beta}$ is a $(p+1)$-vector of coefficients.

RSS is defined as:

$$RSS = (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$$

To minimize the RSS set deravative to beta to zero:

$$\frac{\partial RSS}{\partial \boldsymbol{\beta}} = -2\mathbf{X}^T\mathbf{y} + 2\mathbf{X}^T\mathbf{X}\boldsymbol{\beta} = 0$$

$$\mathbf{X}^T\mathbf{X}\boldsymbol{\beta} = \mathbf{X}^T\mathbf{y}$$

Assuming $\mathbf{X}^T\mathbf{X}$ is invertible, we find the following as the minimizer:

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

Question 2 (Linear Regression)

This question relates to the College data set, which can be found in the file `College.csv`. It contains a number of variables for 777 different universities and colleges in the US.

(from the previous HW) Use the `read.csv()` function to read the data into R. Call the loaded data `college`. Make sure that you have the directory set to the correct location for the data.

Before moving on, we're not going to use the college name, so you may remove `X` variable from data.

Also, make sure categorical variables are set as factor variables.

Split your data into two parts: a testing data that contains 100 observations, and the rest as training data. You may use `sample` function to get the indices of the testing data. For this question, you need to set a random seed while generating this split so that the result can be replicated. Use 4322 as the random seed. Report the mean of `Outstate` of your testing data and training data, respectively.

```
college <- read.csv("College.csv")
college <- college[ , !(names(college) %in% "X")]

# factorizing the categirical features ( only private)
college$Private <- as.factor(college$Private)
set.seed(4322)
summary(college)
```

```
##  Private        Apps           Accept          Enroll       Top10perc
##  No :212   Min.   :   81   Min.   :   72   Min.   :  35   Min.   : 1.00
##  Yes:565   1st Qu.:  776   1st Qu.:  604   1st Qu.: 242   1st Qu.:15.00
##            Median : 1558   Median : 1110   Median : 434   Median :23.00
##            Mean   : 3002   Mean   : 2019   Mean   : 780   Mean   :27.56
##            3rd Qu.: 3624   3rd Qu.: 2424   3rd Qu.: 902   3rd Qu.:35.00
##            Max.   :48094   Max.   :26330   Max.   :6392   Max.   :96.00
##    Top25perc      F.Undergrad     P.Undergrad         Outstate
##  Min.   :  9.0   Min.   :  139   Min.   :    1.0   Min.   : 2340
##  1st Qu.: 41.0   1st Qu.:  992   1st Qu.:   95.0   1st Qu.: 7320
##  Median : 54.0   Median : 1707   Median :  353.0   Median : 9990
##  Mean   : 55.8   Mean   : 3700   Mean   :  855.3   Mean   :10441
##  3rd Qu.: 69.0   3rd Qu.: 4005   3rd Qu.:  967.0   3rd Qu.:12925
##  Max.   :100.0   Max.   :31643   Max.   :21836.0   Max.   :21700
##    Room.Board       Books           Personal         PhD
##  Min.   :1780   Min.   :  96.0   Min.   : 250   Min.   :  8.00
##  1st Qu.:3597   1st Qu.: 470.0   1st Qu.: 850   1st Qu.: 62.00
##  Median :4200   Median : 500.0   Median :1200   Median : 75.00
##  Mean   :4358   Mean   : 549.4   Mean   :1341   Mean   : 72.66
##  3rd Qu.:5050   3rd Qu.: 600.0   3rd Qu.:1700   3rd Qu.: 85.00
##  Max.   :8124   Max.   :2340.0   Max.   :6800   Max.   :103.00
##     Terminal       S.F.Ratio      perc.alumni        Expend
##  Min.   : 24.0   Min.   : 2.50   Min.   : 0.00   Min.   : 3186
##  1st Qu.: 71.0   1st Qu.:11.50   1st Qu.:13.00   1st Qu.: 6751
##  Median : 82.0   Median :13.60   Median :21.00   Median : 8377
##  Mean   : 79.7   Mean   :14.09   Mean   :22.74   Mean   : 9660
##  3rd Qu.: 92.0   3rd Qu.:16.50   3rd Qu.:31.00   3rd Qu.:10830
##  Max.   :100.0   Max.   :39.80   Max.   :64.00   Max.   :56233
##    Grad.Rate
##  Min.   : 10.00
##  1st Qu.: 53.00
```

```
##   Median : 65.00
##   Mean   : 65.46
##   3rd Qu.: 78.00
##   Max.   :118.00
```

```r
# Split
test_indices <- sample(1:nrow(college), 100)
college_test <- college[test_indices, ]
college_train <- college[-test_indices, ]

# Report the mean
mean_test <- mean(college_test$Outstate)
mean_train <- mean(college_train$Outstate)

mean_test
```

```
## [1] 9955.46
```

```r
mean_train
```

```
## [1] 10512.34
```

(a) Now, split your training data into two parts: validation data (100 observations), and the rest as estimation data. Use the random seed 4323.

```r
set.seed(4323)

# Split training data
validation_indices <- sample(1:nrow(college_train), 100)
college_validation <- college_train[validation_indices, ]
college_estimation <- college_train[-validation_indices, ]
```

(b) We're interested in predicting `Enroll`. First, run the linear regression on the estimation data including all variables. What is the feature variable with the highest p-value?

```r
# Fit linear model
fit_all <- lm(Enroll ~ ., data = college_estimation)

summary(fit_all)
```

```
##
## Call:
## lm(formula = Enroll ~ ., data = college_estimation)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1321.94   -61.33   -10.51    49.30  1628.37
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept) 127.702970  93.789853   1.362 0.173877
```

```
## PrivateYes  -41.624379  32.844253  -1.267 0.205566    
## Apps         -0.052226   0.008104  -6.444 2.51e-10 ***
## Accept        0.195585   0.014194  13.779  < 2e-16 ***
## Top10perc     5.049148   1.360485   3.711 0.000227 ***
## Top25perc    -2.906129   1.056250  -2.751 0.006127 ** 
## F.Undergrad   0.127673   0.004805  26.569  < 2e-16 ***
## P.Undergrad  -0.018807   0.008618  -2.182 0.029507 *  
## Outstate     -0.005375   0.004428  -1.214 0.225312    
## Room.Board   -0.016068   0.011220  -1.432 0.152666    
## Books        -0.009285   0.056086  -0.166 0.868577    
## Personal      0.010304   0.014305   0.720 0.471649    
## PhD          -0.269688   1.087562  -0.248 0.804245    
## Terminal     -0.360318   1.199981  -0.300 0.764083    
## S.F.Ratio     1.281315   3.066507   0.418 0.676223    
## perc.alumni   1.489767   0.925684   1.609 0.108100    
## Expend        0.006431   0.003387   1.899 0.058080 .  
## Grad.Rate     0.790790   0.673169   1.175 0.240603    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 206 on 559 degrees of freedom
## Multiple R-squared:  0.9531, Adjusted R-squared:  0.9517 
## F-statistic: 668.5 on 17 and 559 DF,  p-value: < 2.2e-16
```

```r
highest_p_value <- max(coef(summary(fit_all))[ , "Pr(>|t|)"])
highest_p_variable <- names(which(coef(summary(fit_all))[ , "Pr(>|t|)"] == highest_p_value))

highest_p_variable
```

```
## [1] "Books"
```

(c) Run the regression again, but this time, without that variable (with the highest p-value from previous regression) and find the feature variable with the highest p-value with the highest p-value in the new regression. Repeat this step until all the variables have p-value less than 0.1.

```r
#first we exclude the books and enrolls
current_variables <- names(college_estimation)
current_variables <- current_variables[current_variables != "Enroll"]

current_variables <- setdiff(current_variables, "Books")

str(college$Private)
```

```
##  Factor w/ 2 levels "No","Yes": 2 2 2 2 2 2 2 2 2 2 ...
```

```r
# Function to fit the model then get the variable with the highest p
get_highest_p_value_variable <- function(data, variables) {
  formula <- as.formula(paste("Enroll ~", paste(variables, collapse = " + ")))
  fit <- lm(formula, data = data)
  p_values <- summary(fit)$coefficients[-1, "Pr(>|t|)"]

  if (length(p_values) == 0) return(list(variable = NULL, max_p_value = NA, model = fit))
```

```r
  max_p_value <- max(p_values)
  highest_p_variable <- names(which(p_values == max_p_value))

  # Handle the factor 'Private' correctly, it gives error because of the previous factorizeing function
  if ("PrivateYes" %in% highest_p_variable) {
    highest_p_variable <- "Private"
  }

  return(list(variable = highest_p_variable, max_p_value = max_p_value, model = fit))
}

max_iterations <- length(current_variables)
iterations <- 0

repeat {
  result <- get_highest_p_value_variable(college_estimation, current_variables)
  if (is.null(result$variable) || result$max_p_value <= 0.1) {
    final_model <- result$model
    break
  } else {
    current_variables <- setdiff(current_variables, result$variable)
    print(paste("Removing variable:", result$variable))  # Debugging print to see progress
  }

  iterations <- iterations + 1
  if (iterations > max_iterations) {
    print("Reached maximum iterations.")
    break
  }
}
```

```
## [1] "Removing variable: PhD"
## [1] "Removing variable: S.F.Ratio"
## [1] "Removing variable: Personal"
## [1] "Removing variable: Terminal"
## [1] "Removing variable: Grad.Rate"
## [1] "Removing variable: Private"
## [1] "Removing variable: Room.Board"
```

```r
# Summary of the final model
summary(final_model)
```

```
##
## Call:
## lm(formula = formula, data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1341.15   -61.55   -13.05    52.49  1632.88
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept) 109.968345  39.697446   2.770  0.00579 **
## Apps         -0.051656   0.007847  -6.583 1.05e-10 ***
## Accept        0.193350   0.013952  13.859  < 2e-16 ***
## Top10perc     5.270817   1.340517   3.932 9.47e-05 ***
## Top25perc    -3.057038   1.027394  -2.976  0.00305 **
## F.Undergrad   0.130682   0.004449  29.371  < 2e-16 ***
## P.Undergrad  -0.021442   0.008365  -2.563  0.01063 *
## Outstate     -0.009877   0.003537  -2.793  0.00540 **
## perc.alumni   1.677391   0.879825   1.907  0.05709 .
## Expend        0.005032   0.002964   1.698  0.09006 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 205.7 on 567 degrees of freedom
## Multiple R-squared:  0.9526, Adjusted R-squared:  0.9519
## F-statistic:  1267 on 9 and 567 DF,  p-value: < 2.2e-16
```

(d) Find validation MSE of all the models in (b) and (c). Report the model with the smallest validation MSE.

```r
calculate_mse <- function(model, data) {
  predictions <- predict(model, newdata = data)
  mse <- mean((data$Enroll - predictions)^2)
  return(mse)
}

#  model with all variables
mse_all <- calculate_mse(fit_all, college_validation)

#  final model from sec c
mse_final <- calculate_mse(final_model, college_validation)


mse_all
```

```
## [1] 33315.36
```

```r
mse_final
```

```
## [1] 33588.9
```

```r
best_model <- ifelse(mse_final < mse_all, "Final Model from sec c", " Model with All Variables")

best_model
```

```
## [1] " Model with All Variables"
```

(e) Report your test MSE of your chosen model in part (d).

```r
chosen_model <- if(best_model == "Final Model from sec c") final_model else fit_all

test_mse <- calculate_mse(chosen_model, college_test)
test_mse
```

```
## [1] 46148.68
```

Question 3 (k-NN)

This question should be answered using the `Carseats` data set form `ISLR2` package.

Make sure all categorical variables are set as factor variables, and omit any missing data.

```r
data("Carseats")

str(Carseats)
```

```
## 'data.frame':    400 obs. of  11 variables:
##  $ Sales      : num  9.5 11.22 10.06 7.4 4.15 ...
##  $ CompPrice  : num  138 111 113 117 141 124 115 136 132 132 ...
##  $ Income     : num  73 48 35 100 64 113 105 81 110 113 ...
##  $ Advertising: num  11 16 10 4 3 13 0 15 0 0 ...
##  $ Population : num  276 260 269 466 340 501 45 425 108 131 ...
##  $ Price      : num  120 83 80 97 128 72 108 120 124 124 ...
##  $ ShelveLoc  : Factor w/ 3 levels "Bad","Good","Medium": 1 2 3 3 1 1 3 2 3 3 ...
##  $ Age        : num  42 65 59 55 38 78 71 67 76 76 ...
##  $ Education  : num  17 10 12 14 13 16 15 10 10 17 ...
##  $ Urban      : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 1 2 2 1 1 ...
##  $ US         : Factor w/ 2 levels "No","Yes": 2 2 2 2 1 2 1 2 1 2 ...
```

```r
# Convert categirical variables
Carseats$ShelveLoc <- as.factor(Carseats$ShelveLoc)
Carseats$Urban <- as.factor(Carseats$Urban)
Carseats$US <- as.factor(Carseats$US)

summary(Carseats)
```

```
##      Sales          CompPrice       Income        Advertising    
##  Min.   : 0.000   Min.   : 77   Min.   : 21.00   Min.   : 0.000  
##  1st Qu.: 5.390   1st Qu.:115   1st Qu.: 42.75   1st Qu.: 0.000  
##  Median : 7.490   Median :125   Median : 69.00   Median : 5.000  
##  Mean   : 7.496   Mean   :125   Mean   : 68.66   Mean   : 6.635  
##  3rd Qu.: 9.320   3rd Qu.:135   3rd Qu.: 91.00   3rd Qu.:12.000  
##  Max.   :16.270   Max.   :175   Max.   :120.00   Max.   :29.000  
##    Population        Price        ShelveLoc         Age          Education   
##  Min.   : 10.0   Min.   : 24.0   Bad   : 96   Min.   :25.00   Min.   :10.0  
##  1st Qu.:139.0   1st Qu.:100.0   Good  : 85   1st Qu.:39.75   1st Qu.:12.0  
##  Median :272.0   Median :117.0   Medium:219   Median :54.50   Median :14.0  
##  Mean   :264.8   Mean   :115.8                Mean   :53.32   Mean   :13.9  
##  3rd Qu.:398.5   3rd Qu.:131.0                3rd Qu.:66.00   3rd Qu.:16.0  
##  Max.   :509.0   Max.   :191.0                Max.   :80.00   Max.   :18.0  
##  Urban       US     
##  No :118   No :142  
##  Yes:282   Yes:258  
##                     
##                     
##                     
## 
```

(a) Set 10% of whole data as a test set, and the rest as a training set. Split the training set into validation set (10% of training data) and the rest of the training set as a estimation set. Use the random seed `4324`.

```r
set.seed(4324)

# Split data
n <- nrow(Carseats)
test_indices <- sample(1:n, size = round(0.1 * n))
carseats_test <- Carseats[test_indices, ]
carseats_train <- Carseats[-test_indices, ]



train_indices <- sample(1:nrow(carseats_train), size = round(0.9 * nrow(carseats_train)))
carseats_estimation <- carseats_train[train_indices, ]
carseats_validation <- carseats_train[-train_indices, ]

# Check the sizee
cat("Test set size:", nrow(carseats_test), "\n")
```

```
## Test set size: 40
```

```r
cat("Training set size:", nrow(carseats_train), "\n")
```

```
## Training set size: 360
```

```r
cat("Estimation set size:", nrow(carseats_estimation), "\n")
```

```
## Estimation set size: 324
```

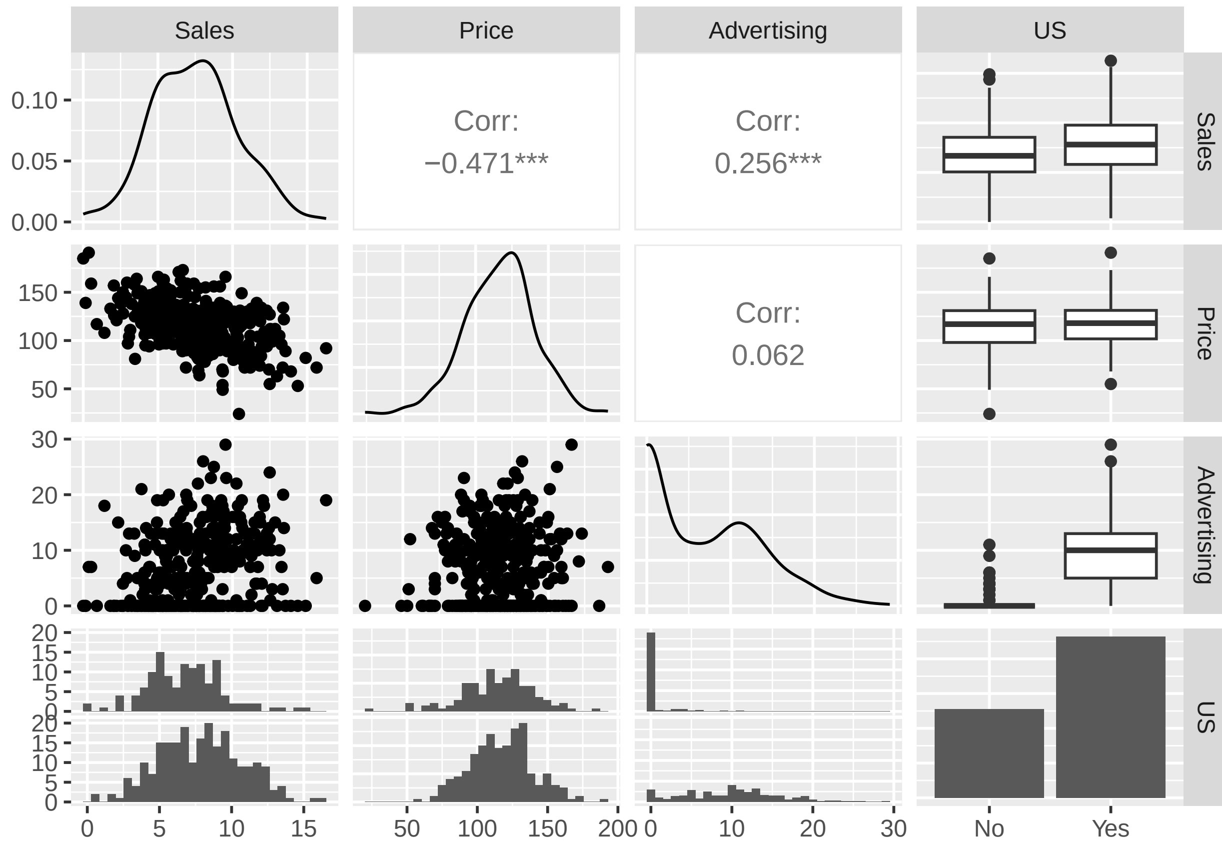
```r
cat("Validation set size:", nrow(carseats_validation), "\n")
```

```
## Validation set size: 36
```

(b) Conduct the EDA on the training set.

```r
ggpairs(carseats_train, columns = c("Sales", "Price", "Advertising", "US"))
```

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




```r
summary(carseats_train)
```

```
##      Sales          CompPrice         Income        Advertising
##  Min.   : 0.000   Min.   : 77.0   Min.   : 21.00   Min.   : 0.000
##  1st Qu.: 5.345   1st Qu.:115.0   1st Qu.: 43.75   1st Qu.: 0.000
##  Median : 7.510   Median :125.0   Median : 69.00   Median : 5.000
##  Mean   : 7.495   Mean   :125.3   Mean   : 68.95   Mean   : 6.575
##  3rd Qu.: 9.320   3rd Qu.:135.0   3rd Qu.: 91.00   3rd Qu.:11.250
##  Max.   :16.270   Max.   :175.0   Max.   :120.00   Max.   :29.000
##    Population        Price        ShelveLoc        Age          Education
##  Min.   : 10.0   Min.   : 24.0   Bad   : 85   Min.   :25.00   Min.   :10.00
##  1st Qu.:136.2   1st Qu.:100.8   Good  : 77   1st Qu.:39.00   1st Qu.:12.00
##  Median :268.5   Median :118.0   Medium:198   Median :54.00   Median :14.00
##  Mean   :262.7   Mean   :116.4                Mean   :52.79   Mean   :13.91
##  3rd Qu.:393.2   3rd Qu.:131.0                3rd Qu.:65.00   3rd Qu.:16.00
##  Max.   :509.0   Max.   :191.0                Max.   :80.00   Max.   :18.00
##  Urban       US
##  No :109   No :128
##  Yes:251   Yes:232
##
##
##
##
```

(c) We're going to fit linear regression models to predict `Sales` using `Price`, `US`, and `Advertising`.

```r
# Define the models
models <- list(
  model1 = Sales ~ Price,
  model2 = Sales ~ US,
  model3 = Sales ~ Advertising,
  model4 = Sales ~ Price + US,
  model5 = Sales ~ US + Advertising,
  model6 = Sales ~ Price + Advertising,
  model7 = Sales ~ Price + US + Advertising,
  model8 = Sales ~ Price * US + Advertising,
  model9 = Sales ~ Price * Advertising + US * Advertising
)

# Fit the models
fitted_models <- lapply(models, function(formula) {
  lm(formula, data = carseats_estimation)
})


lapply(fitted_models, summary)
```

```
## $model1
##
## Call:
## lm(formula = formula, data = carseats_estimation)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.5020 -1.7350 -0.1111  1.5547  7.4694
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 13.910602   0.685815  20.283   <2e-16 ***
## Price       -0.055544   0.005768  -9.629   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.509 on 322 degrees of freedom
## Multiple R-squared:  0.2236, Adjusted R-squared:  0.2212
## F-statistic: 92.72 on 1 and 322 DF,  p-value: < 2.2e-16
##
##
## $model2
##
## Call:
## lm(formula = formula, data = carseats_estimation)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.4229 -1.9254 -0.0779  1.7318  8.4771
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)   6.8199     0.2608   26.15  < 2e-16 ***
## USYes         0.9730     0.3255    2.99  0.00301 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.809 on 322 degrees of freedom
## Multiple R-squared:  0.02701,    Adjusted R-squared:  0.02399
## F-statistic: 8.939 on 1 and 322 DF,  p-value: 0.003007
##
##
## $model3
##
## Call:
## lm(formula = formula, data = carseats_estimation)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.2011 -1.8981 -0.1626  1.6948  8.3439
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.77262    0.21409   31.63  < 2e-16 ***
## Advertising  0.10269    0.02282    4.50  9.5e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.762 on 322 degrees of freedom
## Multiple R-squared:  0.05917,    Adjusted R-squared:  0.05625
## F-statistic: 20.25 on 1 and 322 DF,  p-value: 9.495e-06
##
##
## $model4
##
## Call:
## lm(formula = formula, data = carseats_estimation)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.9459 -1.6341 -0.0128  1.5022  6.9825
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 13.366986   0.679886  19.661  < 2e-16 ***
## Price       -0.057600   0.005639 -10.215  < 2e-16 ***
## USYes        1.219740   0.284194   4.292 2.35e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.444 on 321 degrees of freedom
## Multiple R-squared:  0.2657, Adjusted R-squared:  0.2611
## F-statistic: 58.08 on 2 and 321 DF,  p-value: < 2.2e-16
##
##
## $model5
```

```
##
## Call:
## lm(formula = formula, data = carseats_estimation)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -7.201 -1.898 -0.163  1.695  8.344
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.772970   0.257207  26.333  < 2e-16 ***
## USYes       -0.001071   0.434975  -0.002  0.99804
## Advertising  0.102746   0.031016   3.313  0.00103 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.766 on 321 degrees of freedom
## Multiple R-squared:  0.05917,    Adjusted R-squared:  0.05331
## F-statistic: 10.09 on 2 and 321 DF,  p-value: 5.602e-05
##
##
## $model6
##
## Call:
## lm(formula = formula, data = carseats_estimation)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.9029 -1.6211 -0.0784  1.4224  6.2348
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 13.460354   0.653946  20.583  < 2e-16 ***
## Price       -0.058477   0.005486 -10.659  < 2e-16 ***
## Advertising  0.121005   0.019717   6.137 2.48e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.377 on 321 degrees of freedom
## Multiple R-squared:  0.3051, Adjusted R-squared:  0.3008
## F-statistic: 70.47 on 2 and 321 DF,  p-value: < 2.2e-16
##
##
## $model7
##
## Call:
## lm(formula = formula, data = carseats_estimation)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.8765 -1.5884 -0.0729  1.4541  6.2784
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept) 13.422801   0.662411  20.264  < 2e-16 ***
## Price       -0.058551   0.005497 -10.651  < 2e-16 ***
## USYes        0.140688   0.374562   0.376    0.707
## Advertising  0.114246   0.026713   4.277 2.51e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.38 on 320 degrees of freedom
## Multiple R-squared:  0.3054, Adjusted R-squared:  0.2989
## F-statistic:  46.9 on 3 and 320 DF,  p-value: < 2.2e-16
##
##
## $model8
##
## Call:
## lm(formula = formula, data = carseats_estimation)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.8793 -1.5892 -0.0706  1.4578  6.2890
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 13.3845172  0.9776936  13.690  < 2e-16 ***
## Price       -0.0582140  0.0083782  -6.948 2.09e-11 ***
## USYes        0.2087377  1.3304978   0.157    0.875
## Advertising  0.1142863  0.0267656   4.270 2.58e-05 ***
## Price:USYes -0.0005924  0.0111127  -0.053    0.958
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.384 on 319 degrees of freedom
## Multiple R-squared:  0.3054, Adjusted R-squared:  0.2967
## F-statistic: 35.07 on 4 and 319 DF,  p-value: < 2.2e-16
##
##
## $model9
##
## Call:
## lm(formula = formula, data = carseats_estimation)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.8611 -1.6028 -0.0162  1.4753  6.1163
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)       13.7944346  0.8634248  15.976  < 2e-16 ***
## Price             -0.0611064  0.0073431  -8.322 2.59e-15 ***
## Advertising       -0.1149719  0.1694489  -0.679    0.498
## USYes              0.0159397  0.3939262   0.040    0.968
## Price:Advertising  0.0004444  0.0008204   0.542    0.588
## Advertising:USYes  0.1819881  0.1464572   1.243    0.215
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.381 on 318 degrees of freedom
## Multiple R-squared:  0.3096, Adjusted R-squared:  0.2987
## F-statistic: 28.52 on 5 and 318 DF,  p-value: < 2.2e-16
```

Candidate models:

```
model 1: Sales~Price
model 2: Sales~US
model 3: Sales~Advertising
model 4: Sales~Price+US
model 5: Sales~US+Advertising
model 6: Sales~Price+Advertising
model 7: Sales~Price+US+Price*US
model 8: Sales~US+Advertising+US*Advertising
model 9: Sales~Price+Advertising+Price*Advertising
```

Store all regression models in one list. Run the regressions on the estimation data.

(e) Calculate validation MSE of all models. Choose a single model with the lowest validation MSE.

```r
calculate_mse <- function(model, data) {
  predictions <- predict(model, newdata = data)
  mse <- mean((data$Sales - predictions)^2)
  return(mse)
}


validation_mse <- sapply(fitted_models, calculate_mse, data = carseats_validation)

validation_mse
```

```
##   model1   model2   model3   model4   model5   model6   model7   model8
## 6.794518 7.796904 7.556583 6.190823 7.556891 6.072983 6.050427 6.049539
##   model9
## 6.305631
```

```r
# Identify lowest validation MSE
best_model_index <- which.min(validation_mse)
best_model <- fitted_models[[best_model_index]]
cat("Best model is:", names(models)[best_model_index], "with MSE =", validation_mse[best_model_index],
```

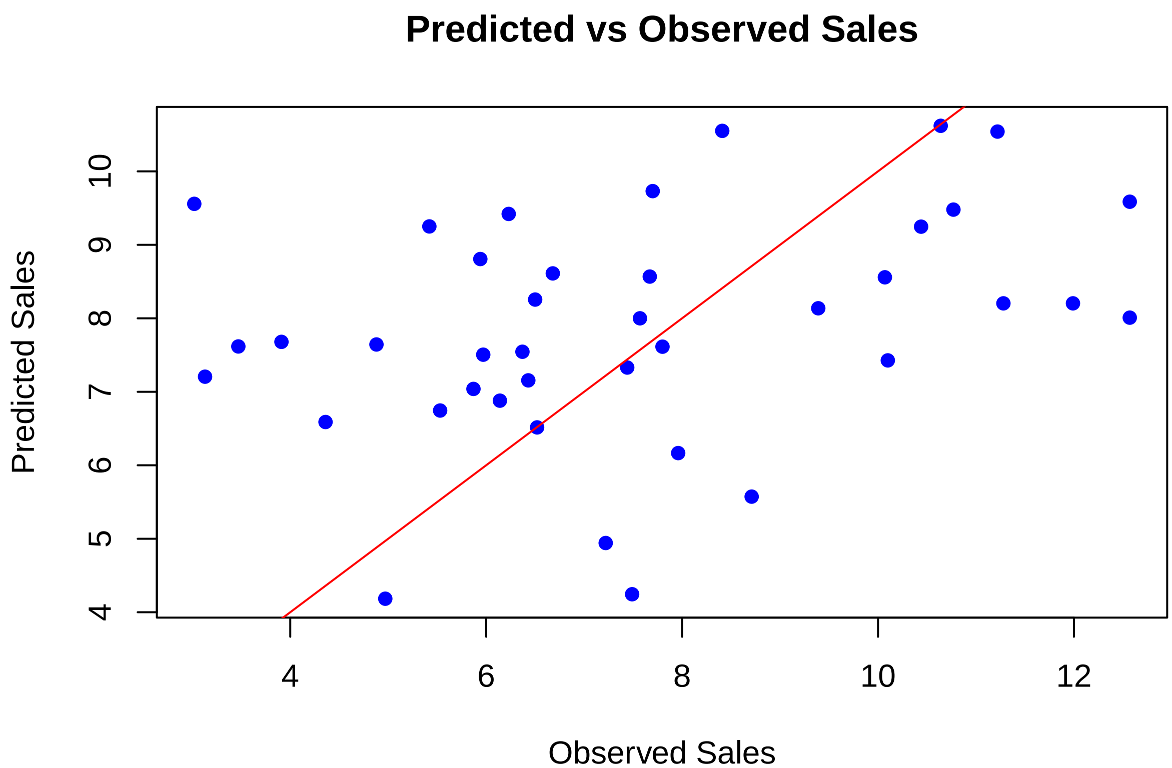
```
## Best model is: model8 with MSE = 6.049539
```

(f) Report your test MSE. Provide a scatter plot of predicted Sales and observed Sales of the test data.

```r
test_mse <- calculate_mse(best_model, carseats_test)
cat("Test MSE for the best model:", test_mse, "\n")
```

```
## Test MSE for the best model: 6.452404
```

```r
test_predictions <- predict(best_model, newdata = carseats_test)
plot(carseats_test$Sales, test_predictions,
     xlab = "Observed Sales", ylab = "Predicted Sales",
     main = "Predicted vs Observed Sales",
     col = "blue", pch = 16)
abline(0, 1, col = "red")
```



Question 4 (k-NN and decision tree)

This question relates to the `Boston` data set of `ISLR2` package.

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.3.3
```

```
## Loading required package: lattice
```

```
library(rpart)
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 4.3.3
```

```
set.seed(432)
trn.idx=sample(1:nrow(ISLR2::Boston),450)
tst.boston=ISLR2::Boston[-trn.idx,]
trn.boston=ISLR2::Boston[trn.idx,]
```

We are splitting the data into two parts: a testing data that contains 56 observations, and the rest 450 observations as training data.

- The goal is to model `medv` (our response variable) with all the other variables in the data.
- In this HW, we'll not worry about scaling variables. We'll tackle that in the future.

(a) Use the following validation-estimation split.

```
set.seed(1)
val.idx=sample(1:nrow(trn.boston),45)
val.boston=trn.boston[val.idx,]
est.boston=trn.boston[-val.idx,]
# Check the sizes of the datasets
cat("Training set size:", nrow(trn.boston), "\n")
```

```
## Training set size: 450
```

```
cat("Estimation set size:", nrow(est.boston), "\n")
```

```
## Estimation set size: 405
```

```
cat("Validation set size:", nrow(val.boston), "\n")
```

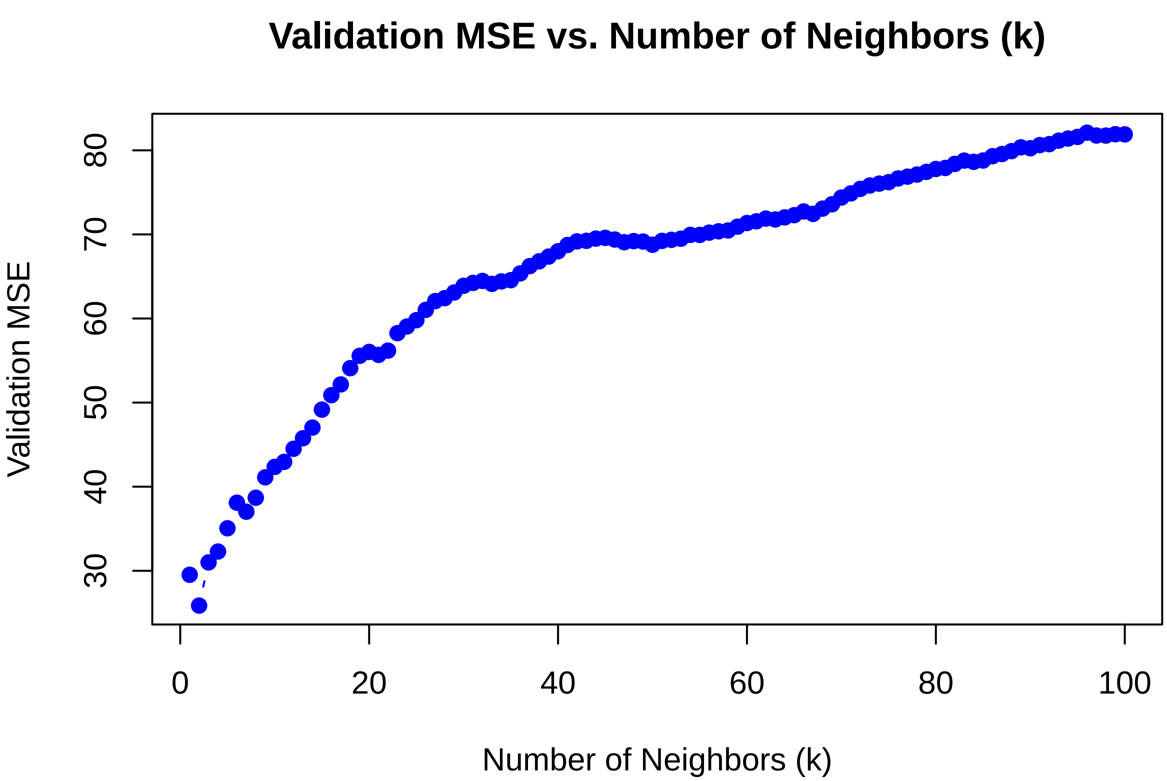
```
## Validation set size: 45
```

```
cat("Test set size:", nrow(tst.boston), "\n")
```

```
## Test set size: 56
```

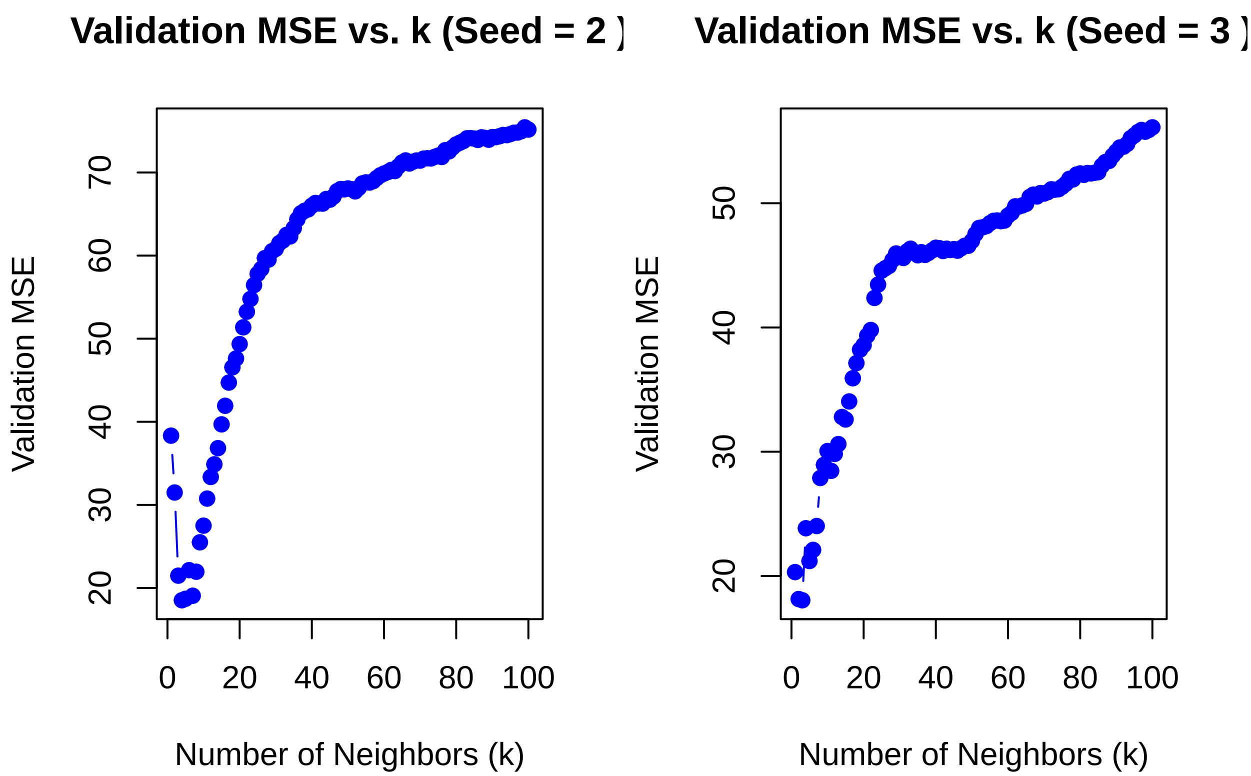
- Use the estimation data and `knnreg` function of `caret` package to perform KNN.
- Train KNN models using values of `k` from 1 to 100 and calculate validation MSE for each `k`.
- Plot the validation MSE versus `k` and show them in the same graph.

```
train_x <- est.boston[, -which(names(est.boston) == "medv")]
train_y <- est.boston$medv
val_x <- val.boston[, -which(names(val.boston) == "medv")]
val_y <- val.boston$medv

# Initialize
k_values <- 1:100
validation_mse <- numeric(length(k_values))

for (k in k_values) {
  knn_model <- knnreg(train_x, train_y, k = k)
  val_predictions <- predict(knn_model, val_x)
  validation_mse[k] <- mean((val_y - val_predictions)^2)
}

# Plot
plot(k_values, validation_mse, type = "b", col = "blue", pch = 19,
     xlab = "Number of Neighbors (k)", ylab = "Validation MSE",
     main = "Validation MSE vs. Number of Neighbors (k)")
```



(b) Repeat (a) with different random seeds, (2,3), and see if your answer changes. If so, why does it change?

```r
knn_analysis_with_seed <- function(seed) {
  set.seed(seed)
  val.idx <- sample(1:nrow(trn.boston), 45)
  val.boston <- trn.boston[val.idx, ]
  est.boston <- trn.boston[-val.idx, ]

  train_x <- est.boston[, -which(names(est.boston) == "medv")]
  train_y <- est.boston$medv
  val_x <- val.boston[, -which(names(val.boston) == "medv")]
  val_y <- val.boston$medv

  validation_mse <- numeric(length(k_values))

  for (k in k_values) {
    knn_model <- knnreg(train_x, train_y, k = k)
    val_predictions <- predict(knn_model, val_x)
    validation_mse[k] <- mean((val_y - val_predictions)^2)
  }

  return(validation_mse)
}
#for other seeds
seeds <- c(2, 3)
mse_results <- lapply(seeds, knn_analysis_with_seed)

# Plot the results
par(mfrow = c(1, 2))
for (i in 1:length(seeds)) {
  plot(k_values, mse_results[[i]], type = "b", col = "blue", pch = 19,
       xlab = "Number of Neighbors (k)", ylab = "Validation MSE",
       main = paste("Validation MSE vs. k (Seed =", seeds[i], ")"))
}
```



Answer: The validation MSE changes with different random seeds becuz each seed results in a diffrent random split of the data. This means that the estimation and validation sets are different, which effects how the kNN model is trained and evaluted. The differnces in data subsets lead to variations in model performance and validation MSE. This variablity is typical in machine learning and show the importance of using consistent data splits or multiple runs to assess model stablity.

(c) Use the estimation/validation data from (a) with random seed (1) and `rpart` and `rpart.plot`function to perform decision tree.

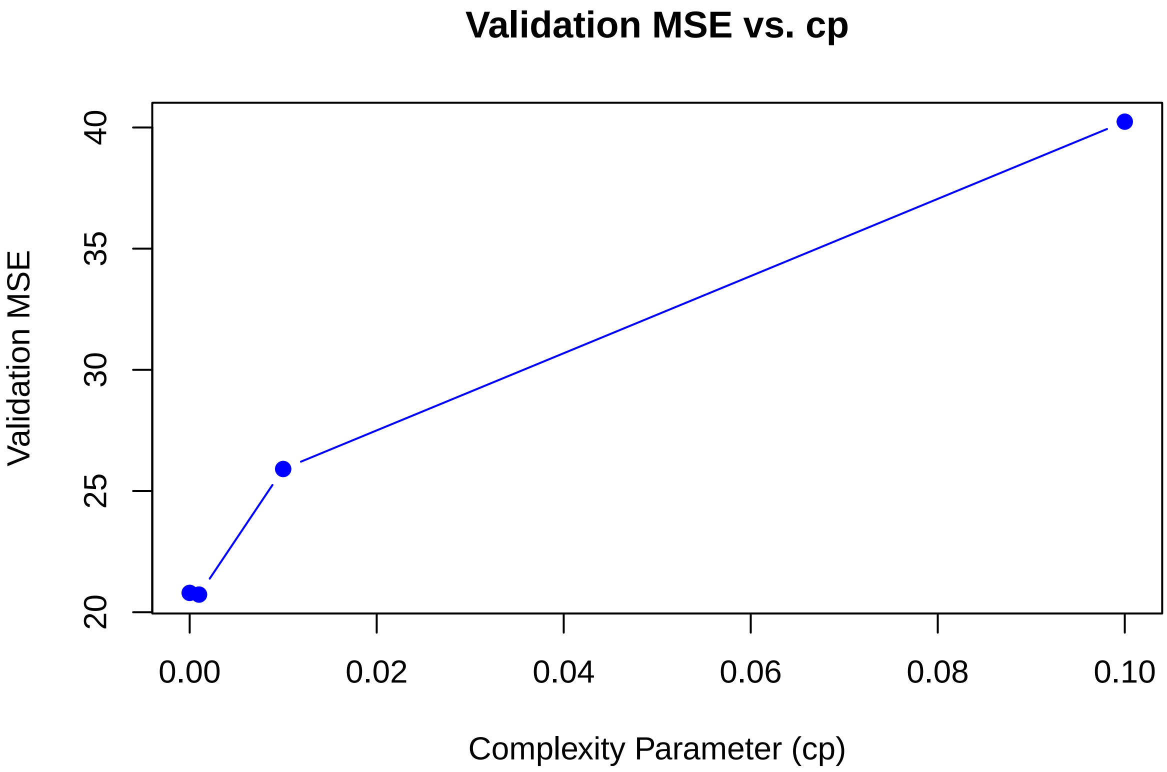
- Start with default setting of R.
- Train decision tree models using cp=0, 0.001, 0.01, 0.1.
- Students may explore other tuning parameters as needed.
- Show your tree results using `rpart.plot` function.
- Compute validation MSE versus different cp values.
- Choose cp with lowest validation MSE.

```
cp_values <- c(0, 0.001, 0.01, 0.1)


models <- list()
validation_mse <- numeric(length(cp_values))

# Train models with diff cp
for (i in 1:length(cp_values)) {
  cp <- cp_values[i]
  model <- rpart(medv ~ ., data = est.boston, control = rpart.control(cp = cp))
  models[[i]] <- model
  val_predictions <- predict(model, newdata = val.boston)
  validation_mse[i] <- mean((val.boston$medv - val_predictions)^2)
```

```r
  cat("cp =", cp, "- Validation MSE:", validation_mse[i], "\n")
}
```

```
## cp = 0 - Validation MSE: 20.79615
## cp = 0.001 - Validation MSE: 20.72719
## cp = 0.01 - Validation MSE: 25.90849
## cp = 0.1 - Validation MSE: 40.23923
```

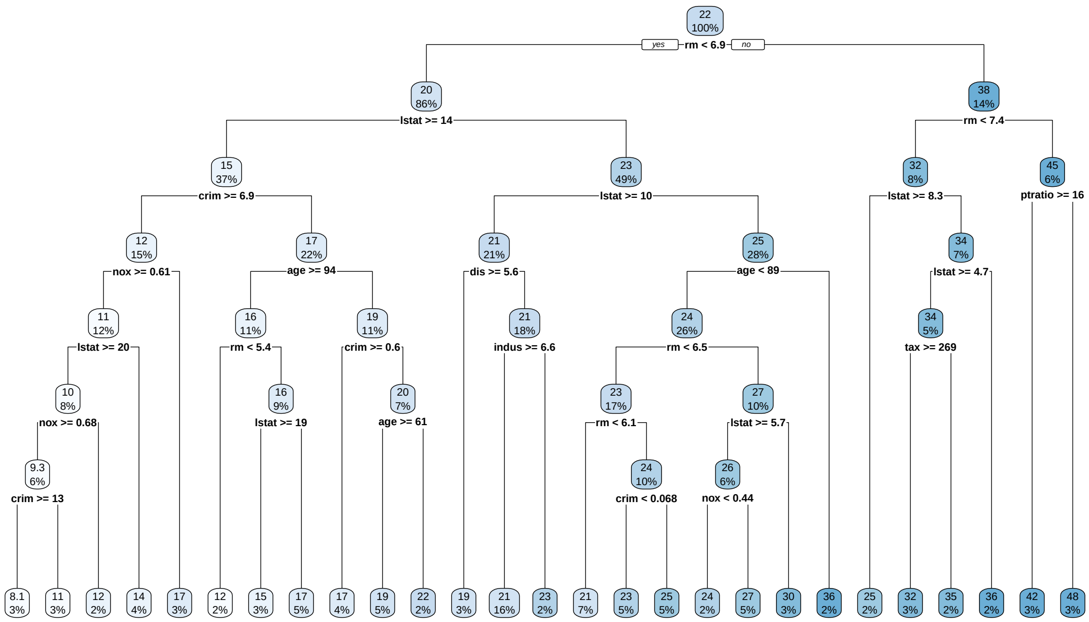
```r
# Plot
plot(cp_values, validation_mse, type = "b", col = "blue", pch = 19,
     xlab = "Complexity Parameter (cp)", ylab = "Validation MSE",
     main = "Validation MSE vs. cp")
```




```r
# Identify and print the best cp value
best_model_index <- which.min(validation_mse)
best_cp <- cp_values[best_model_index]
cat("The best cp value is", best_cp, "with the lowest validation MSE of", validation_mse[best_model_inde
```

```
## The best cp value is 0.001 with the lowest validation MSE of 20.72719
```

```r
# Display the tree
best_model <- models[[best_model_index]]
rpart.plot(best_model)
```

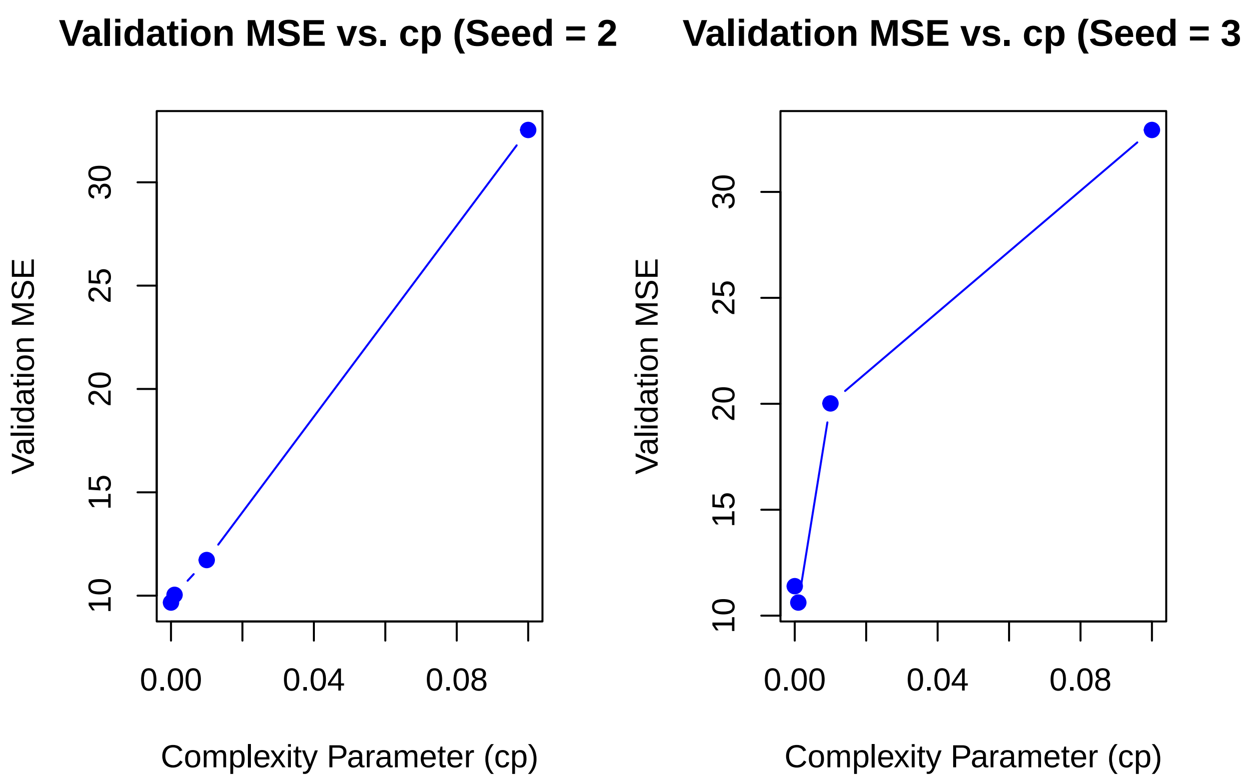


(d) Repeat (c) with estimation/validation set with different random seeds, (2,3), and see if your answer changes. If so, why does it change?

```
decision_tree_analysis_with_seed <- function(seed) {
  set.seed(seed)
  val.idx <- sample(1:nrow(trn.boston), 45)
  val.boston <- trn.boston[val.idx, ]
  est.boston <- trn.boston[-val.idx, ]

  validation_mse <- numeric(length(cp_values))

  for (i in 1:length(cp_values)) {
    cp <- cp_values[i]
    model <- rpart(medv ~ ., data = est.boston, control = rpart.control(cp = cp))
    val_predictions <- predict(model, newdata = val.boston)
    validation_mse[i] <- mean((val.boston$medv - val_predictions)^2)

    # Print the MSE for each cp
    cat("Seed =", seed, "cp =", cp, "- Validation MSE:", validation_mse[i], "\n")
  }

  # Identify the best cp
  best_cp_index <- which.min(validation_mse)
  best_cp <- cp_values[best_cp_index]
  best_mse <- validation_mse[best_cp_index]

  cat("For seed =", seed, "the best cp value is", best_cp, "with the lowest validation MSE of", best_ms

  return(validation_mse)
}

# Calculate val MSEs for diff seeds
seeds <- c(2, 3)
mse_results <- lapply(seeds, decision_tree_analysis_with_seed)
```

```
## Seed = 2 cp = 0 - Validation MSE: 9.666538
## Seed = 2 cp = 0.001 - Validation MSE: 10.03971
## Seed = 2 cp = 0.01 - Validation MSE: 11.72382
## Seed = 2 cp = 0.1 - Validation MSE: 32.53138
## For seed = 2 the best cp value is 0 with the lowest validation MSE of 9.666538
## Seed = 3 cp = 0 - Validation MSE: 11.39089
## Seed = 3 cp = 0.001 - Validation MSE: 10.61798
## Seed = 3 cp = 0.01 - Validation MSE: 20.01785
## Seed = 3 cp = 0.1 - Validation MSE: 32.92429
## For seed = 3 the best cp value is 0.001 with the lowest validation MSE of 10.61798
```

```r
# Plot the results
par(mfrow = c(1, 2))
for (i in 1:length(seeds)) {
  plot(cp_values, mse_results[[i]], type = "b", col = "blue", pch = 19,
       xlab = "Complexity Parameter (cp)", ylab = "Validation MSE",
       main = paste("Validation MSE vs. cp (Seed =", seeds[i], ")"))
}
```



In this ML methods such as the KNN the results changed because of having different testing data. As you can see even the best cp can be different for different random seed numbers as in 1 and 3 the best is cp=0.001 but in the random see =2 it has ben the cp=0. However, you can see the change between the cp=0.001 and cp = 0 is not a lot so it can change by changing the random seed.