# TakeHomeAug4

2024-07-23

## Take-home #1 (Chapter 2 #10)

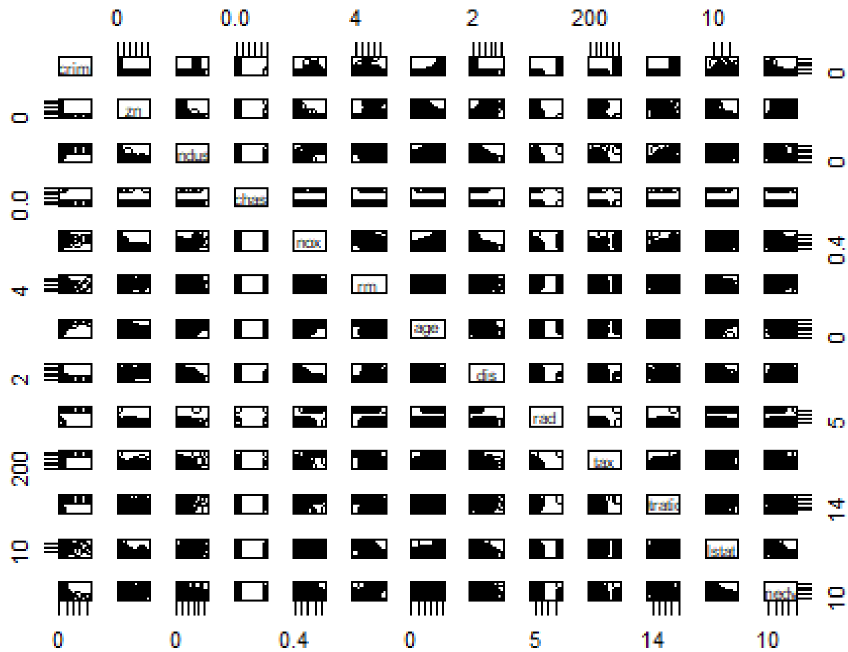
### Part A

```
#Access Boston Data
library(ISLR2)
my_data = Boston
#Dimensions of table
rows_columns = dim(my_data)
#Print dimensions
nrow(Boston)

## [1] 506

ncol(Boston)

## [1] 13
```

Answer: 506 rows and 13 variables

### Part B

```
#Convert the 'chas' and 'rad' column to numeric type
my_data$chas <- as.numeric(my_data$chas)
my_data$rad <- as.numeric(my_data$rad)
# Create a scatterplot matrix of the variables in the my_data dataframe
pairs(my_data)
```




Answer: The plot itself doesn't really help us reach any conclusions. What it does tell us is that some variables may be correlated. For example, we can see that crim and dis may seems to have a relationship.

## Part C

```
library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

#Initialize empty lists to store results
i <- character()
j <- character()
cor <- numeric()
p <- numeric()

#Loop through each variable in the my_data
for (var in names(my_data)) {
  if (var != "crim") {
```

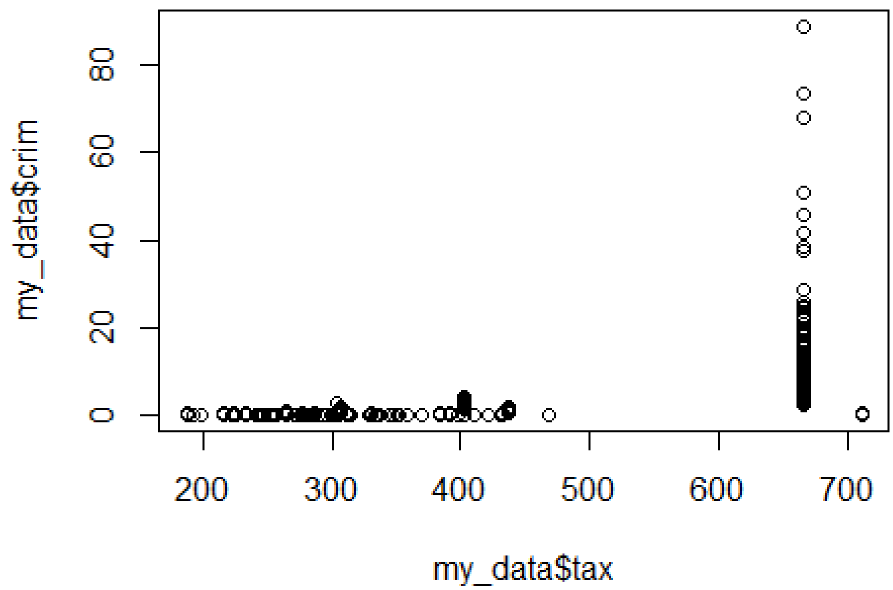
```r
    # Calculate correlation and p-value
    test <- cor.test(my_data$crim, my_data[[var]])
    # Store results
    i <- c(i, "crim")
    j <- c(j, var)
    cor <- c(cor, test$estimate)
    p <- c(p, test$p.value)
  }
}

#Combine results into a my_data
results <- data.frame(i, j, cor, p)

#Display the results
print(results)
```

```
##       i       j         cor            p
## 1  crim      zn -0.20046922 5.506472e-06
## 2  crim   indus  0.40658341 1.450349e-21
## 3  crim    chas -0.05589158 2.094345e-01
## 4  crim     nox  0.42097171 3.751739e-23
## 5  crim      rm -0.21924670 6.346703e-07
## 6  crim     age  0.35273425 2.854869e-16
## 7  crim     dis -0.37967009 8.519949e-19
## 8  crim     rad  0.62550515 2.693844e-56
## 9  crim     tax  0.58276431 2.357127e-47
## 10 crim ptratio  0.28994558 2.942922e-11
## 11 crim   lstat  0.45562148 2.654277e-27
## 12 crim    medv -0.38830461 1.173987e-19
```

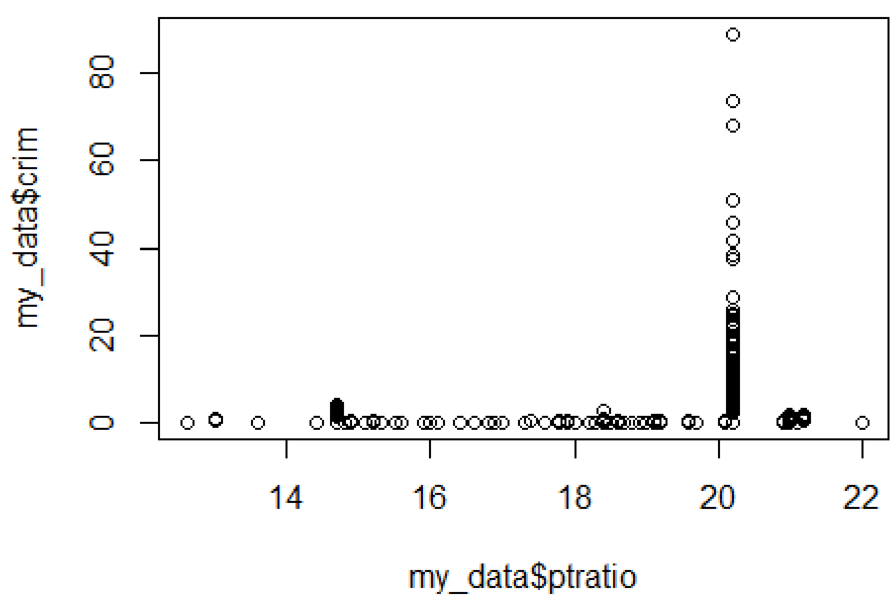
Answer: To see the relationships between the variables and crime rate, I made a correlation matrix with p-values. The correlation matrix shows that there is indeed a relationship between the predictors and crime rate. For example, rad and crim has a correlation coefficient of 0.625 and a really low p-value indicating a strong relationship. Some variables demonstrate a negative correlation too.

## Part D

```r
#Scatter Plots for tax  and ptration against crime
plot(my_data$tax,my_data$crim)
```



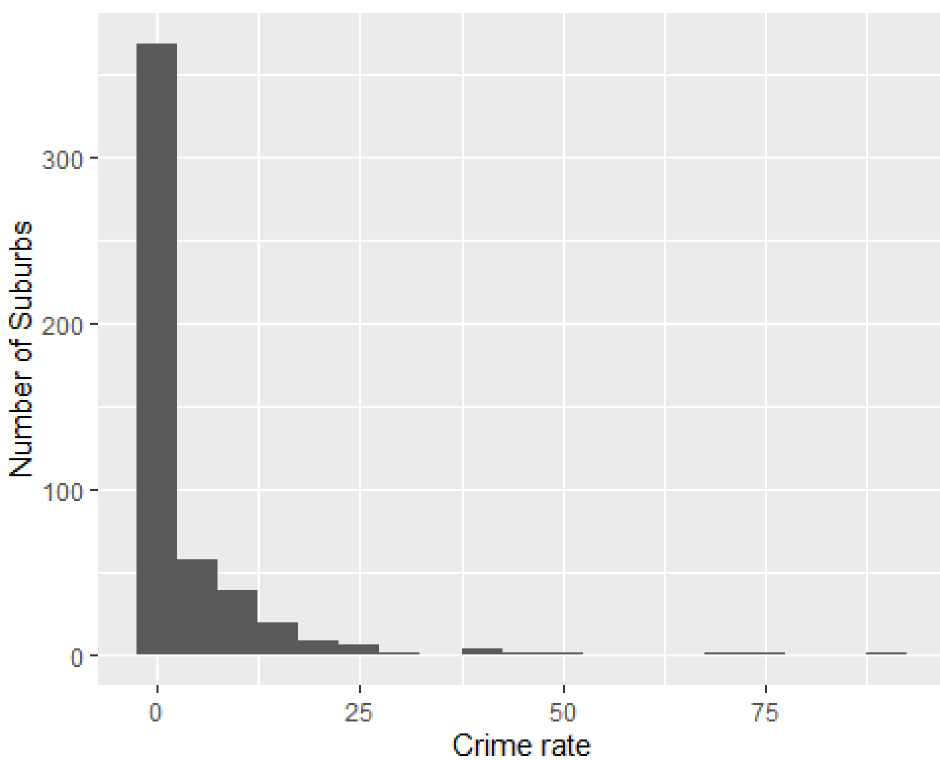

```
plot(my_data$ptratio,my_data$crim)
```



```r
#Ranges Displayed for
range(my_data$tax)

## [1] 187 711

range(my_data$ptratio)

## [1] 12.6 22.0
```

- The range for tax is 187 - 711 and The range for ptratio is 12.6 - 22.0
- Particularly the census tracts where tax is 666 and ptratio is 20.2, we see a spike in the range of crime rate. However this does not mean that the areas with the highest tax rate have the highest crime rates. Perhaps because some properties at nicer places have higher tax rates for example comparing NY city to the Hamptons.

```r
#Plot Histogram for Crime Rate Across Different Suburbs
library(ggplot2)
qplot(my_data$crim, binwidth=5, xlab= "Crime rate", ylab= "Number of
Suburbs")

## Warning: `qplot()` was deprecated in ggplot2 3.4.0.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```




- From the above figure, we can see that while some neighborhoods have low crime rates, others have extremely high rates. Additionally, given that the median crime rate is 0.26% and the maximum is 89%, it is evident that certain neighborhoods experience alarmingly high crime rates.

## Part E

```
# Census tracts in this data set bound the Charles river
sum(my_data$chas)

## [1] 35
```

Answer: Number of Census Tracts that bound Charles River is 35

## Part F

```
#Median pupil-teacher ratio
median(my_data$ptratio)

## [1] 19.05
```

Answer: Median PT ratio is 19.05

## Part G

```
#Find the census tract with lowest median value of owner occupied homes
ordered_by_medv <- my_data[order(my_data$medv),]
ordered_by_medv [1, ]

##        crim zn indus chas   nox    rm age    dis rad tax ptratio lstat
medv
## 399 38.3518  0  18.1    0 0.693 5.453 100 1.4896  24 666    20.2 30.59
5

#Find Summary of all columns of data set so we can compare
summary(ordered_by_medv)

##       crim                zn             indus            chas
##  Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   Min.   :0.00000
##  1st Qu.: 0.08205   1st Qu.:  0.00   1st Qu.: 5.19   1st Qu.:0.00000
##  Median : 0.25651   Median :  0.00   Median : 9.69   Median :0.00000
##  Mean   : 3.61352   Mean   : 11.36   Mean   :11.14   Mean   :0.06917
##  3rd Qu.: 3.67708   3rd Qu.: 12.50   3rd Qu.:18.10   3rd Qu.:0.00000
##  Max.   :88.97620   Max.   :100.00   Max.   :27.74   Max.   :1.00000
##       nox               rm             age              dis
##  Min.   :0.3850   Min.   :3.561   Min.   :  2.90   Min.   : 1.130
##  1st Qu.:0.4490   1st Qu.:5.886   1st Qu.: 45.02   1st Qu.: 2.100
##  Median :0.5380   Median :6.208   Median : 77.50   Median : 3.207
##  Mean   :0.5547   Mean   :6.285   Mean   : 68.57   Mean   : 3.795
##  3rd Qu.:0.6240   3rd Qu.:6.623   3rd Qu.: 94.08   3rd Qu.: 5.188
##  Max.   :0.8710   Max.   :8.780   Max.   :100.00   Max.   :12.127
##       rad              tax           ptratio          lstat
##  Min.   : 1.000   Min.   :187.0   Min.   :12.60   Min.   : 1.73
##  1st Qu.: 4.000   1st Qu.:279.0   1st Qu.:17.40   1st Qu.: 6.95
##  Median : 5.000   Median :330.0   Median :19.05   Median :11.36
##  Mean   : 9.549   Mean   :408.2   Mean   :18.46   Mean   :12.65
```

```
##  3rd Qu.:24.000   3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:16.95
##  Max.   :24.000   Max.   :711.0   Max.   :22.00   Max.   :37.97
##       medv
##  Min.   : 5.00
##  1st Qu.:17.02
##  Median :21.20
##  Mean   :22.53
##  3rd Qu.:25.00
##  Max.   :50.00
```

- Crime is very high(38.35) falling even higher than the third quartile(3.67) when compared to overall range.
- No residential land zones for lots over 25,000 sq.ft.
- Proprotion of non-retail business acres per town is very high falling at just on the third quartile when compared to overall range.
- Suburb 399 does not bound the Charles River.
- Nox is very high (0.693) falling even higher than third third quartile (0.6240) when compared to overall range.
- Rooms per dwelling metric falls under the lower category (5.443), lower than the 1st quartile ( 5.886).
- One of the highest undefined proportion of owner occupied unit sbuilt prior to 1940 (100).
- Weighted mean of distances to five Boston employment centres falls at one of the lowest (1.486), lower than the 1st quartile (17.02).
- Index of accessibility to radial highways is the highest when compared to the range of the entire data set.
- Full-value property tax rate per $10,000 is quit high sitting right o the third quartile (666)
- Pupil teacher ration by town is quite high of 20.2 falling on the third qaurtile.
- Lower status of population is quite high (30.59), falling above the third quartiel (16.95).
- Median value of owner occupied homes in $1000s is the lowest in the range (5.00).

## Part H

```
#Do Summation given conditions for rooms per dwelling.
sum(my_data$rm > 7) #64

## [1] 64
```

```r
sum(my_data$rm > 8) #13

## [1] 13
```

Answer:

- 64 suburbs with rooms per dwelling > 7.
- 13 suburbs with roombs per dwelling > 8.

```r
#Find summary for suburbs where rooms per dwelling is greater than 8
summary(filter(my_data, my_data$rm > 8))

##       crim                zn             indus             chas
##  Min.   :0.02009   Min.   : 0.00   Min.   : 2.680   Min.   :0.0000
##  1st Qu.:0.33147   1st Qu.: 0.00   1st Qu.: 3.970   1st Qu.:0.0000
##  Median :0.52014   Median : 0.00   Median : 6.200   Median :0.0000
##  Mean   :0.71879   Mean   :13.62   Mean   : 7.078   Mean   :0.1538
##  3rd Qu.:0.57834   3rd Qu.:20.00   3rd Qu.: 6.200   3rd Qu.:0.0000
##  Max.   :3.47428   Max.   :95.00   Max.   :19.580   Max.   :1.0000
##       nox               rm             age              dis
##  Min.   :0.4161   Min.   :8.034   Min.   : 8.40   Min.   :1.801
##  1st Qu.:0.5040   1st Qu.:8.247   1st Qu.:70.40   1st Qu.:2.288
##  Median :0.5070   Median :8.297   Median :78.30   Median :2.894
##  Mean   :0.5392   Mean   :8.349   Mean   :71.54   Mean   :3.430
##  3rd Qu.:0.6050   3rd Qu.:8.398   3rd Qu.:86.50   3rd Qu.:3.652
##  Max.   :0.7180   Max.   :8.780   Max.   :93.90   Max.   :8.907
##       rad              tax           ptratio           lstat            medv
##  Min.   : 2.000   Min.   :224.0   Min.   :13.00   Min.   :2.47   Min.   
:21.9
##  1st Qu.: 5.000   1st Qu.:264.0   1st Qu.:14.70   1st Qu.:3.32   1st 
Qu.:41.7
##  Median : 7.000   Median :307.0   Median :17.40   Median :4.14   Median 
:48.3
##  Mean   : 7.462   Mean   :325.1   Mean   :16.36   Mean   :4.31   Mean   
:44.2
##  3rd Qu.: 8.000   3rd Qu.:307.0   3rd Qu.:17.40   3rd Qu.:5.12   3rd 
Qu.:50.0
##  Max.   :24.000   Max.   :666.0   Max.   :20.20   Max.   :7.44   Max.   
:50.0
```

- Mean crime rates when compared to the entire data set is much lower when number of dwellings is more than 8.
- There is a low pupil teacher ratio.
- These properties seem to farther away from the highways perhaps because they are not as near to the the centre of the city. Additionally, there are not a lot of non-retail businesses showing that these areas are residential.
- They are older houses.

# Take-home #2 (Chapter 2 #10)

```
#Access Boston Data
library(ISLR2)
p2_data = Boston
```

## Part A

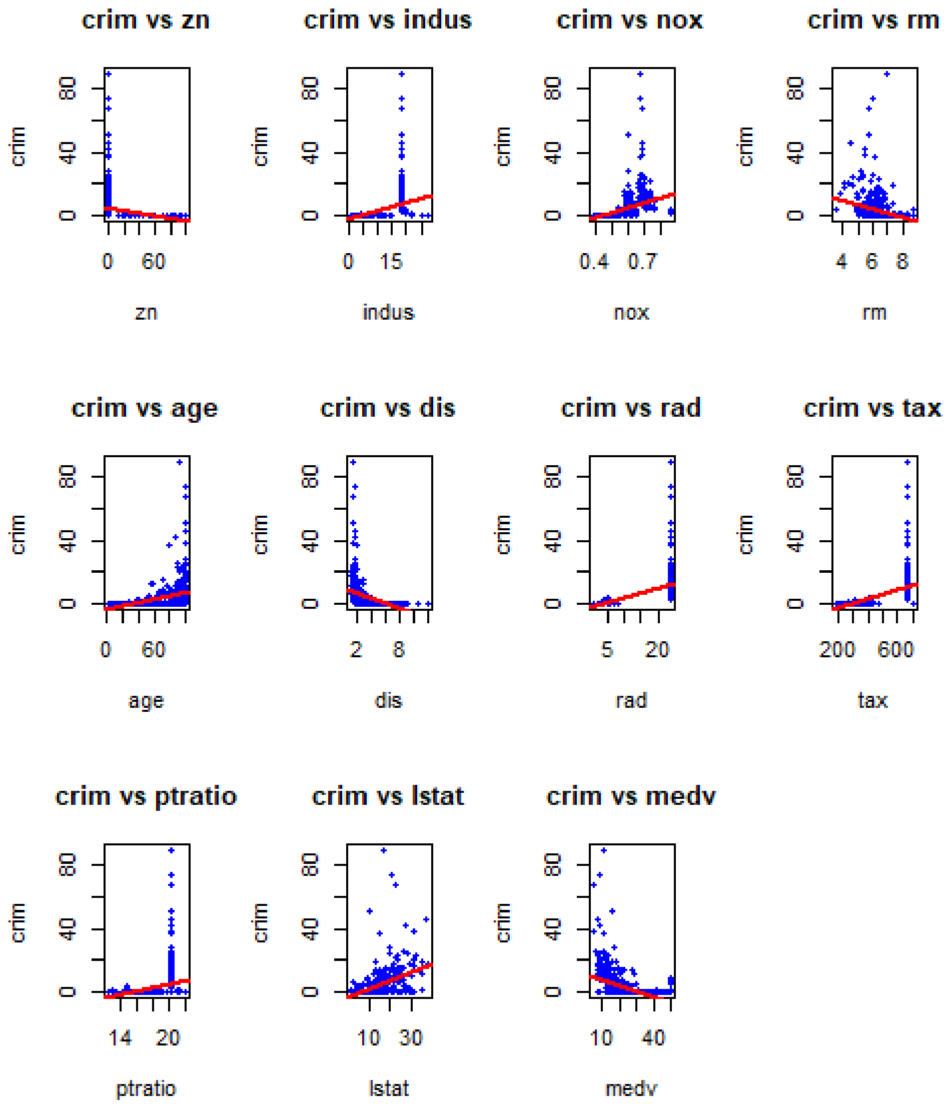
```
#List of all predictors in the dataset except for crim
predictors <- names(Boston)[names(Boston) != "crim"]

#Initialize an empty list to store the model summaries
model_summaries <- list()

#Fit simple linear regression models for each predictor
for (predictor in predictors) {
  formula <- as.formula(paste("crim ~", predictor))
  model <- lm(formula, data = Boston)
  model_summaries[[predictor]] <- summary(model)
}

#Extract significant and non-significant predictors
significant_predictors <- list()
non_significant_predictors <- list()

for (predictor in predictors) {
  coef_summary <- coef(model_summaries[[predictor]])
  p_value <- coef_summary[2, 4]  # p-value of the predictor

  if (p_value < 0.05) {
    significant_predictors <- c(significant_predictors, predictor)
  } else {
    non_significant_predictors <- c(non_significant_predictors, predictor)
  }
}

#Display significant and non-significant predictors
significant_predictors

## [[1]]
## [1] "zn"
##
## [[2]]
## [1] "indus"
##
## [[3]]
## [1] "nox"
##
## [[4]]
## [1] "rm"
##
```

```
## [[5]]
## [1] "age"
##
## [[6]]
## [1] "dis"
##
## [[7]]
## [1] "rad"
##
## [[8]]
## [1] "tax"
##
## [[9]]
## [1] "ptratio"
##
## [[10]]
## [1] "lstat"
##
## [[11]]
## [1] "medv"
```

```
non_significant_predictors
```

```
## [[1]]
## [1] "chas"
```

```
#Set up plot area
par(mfrow = c(2, 4))

#Plot for each significant predictor
for (predictor in significant_predictors) {
  formula <- as.formula(paste("crim ~", predictor))
  model <- lm(formula, data = Boston)

  plot(Boston[[predictor]], Boston$crim, main = paste("crim vs", predictor),
       xlab = predictor, ylab = "crim", pch = 20, col = "blue")
  abline(model, col = "red", lwd = 2)
}
```




#Comments: According to the simple regression model the only predictor that does not have a statistcally significant relationship with crime is chas ( if tract bounds river or not). All other variables seem to have a relationship. The linear lines are present however, it is clear that it is not a very accurate representation of what is going on because of the residuals.

### Part B

```
#Multiple Regression Model with a target variable of Crime
lm.all = lm(crim ~ ., data=p2_data)
summary(lm.all)

##
## Call:
## lm(formula = crim ~ ., data = p2_data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -8.534 -2.248 -0.348  1.087 73.923
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 13.7783938  7.0818258   1.946 0.052271 .
## zn           0.0457100  0.0187903   2.433 0.015344 *
## indus       -0.0583501  0.0836351  -0.698 0.485709
## chas        -0.8253776  1.1833963  -0.697 0.485841
## nox         -9.9575865  5.2898242  -1.882 0.060370 .
## rm           0.6289107  0.6070924   1.036 0.300738
## age         -0.0008483  0.0179482  -0.047 0.962323
## dis         -1.0122467  0.2824676  -3.584 0.000373 ***
## rad          0.6124653  0.0875358   6.997 8.59e-12 ***
## tax         -0.0037756  0.0051723  -0.730 0.465757
## ptratio     -0.3040728  0.1863598  -1.632 0.103393
## lstat        0.1388006  0.0757213   1.833 0.067398 .
## medv        -0.2200564  0.0598240  -3.678 0.000261 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.46 on 493 degrees of freedom
## Multiple R-squared:  0.4493, Adjusted R-squared:  0.4359
## F-statistic: 33.52 on 12 and 493 DF,  p-value: < 2.2e-16
```

#Comment: Accordng to the table with he p-values listed for each predictor zn, dis, rad, and medv have a p-value lower than 0.05, meaning they are the only variables that demonstrate a statistically significant relationship within the multiple regression model.

### Part C

```
#List of all predictors in the dataset except for crim
predictors <- names(Boston)[names(Boston) != "crim"]

#Initialize vectors to store coefficients
univariate_coeffs <- c()
```

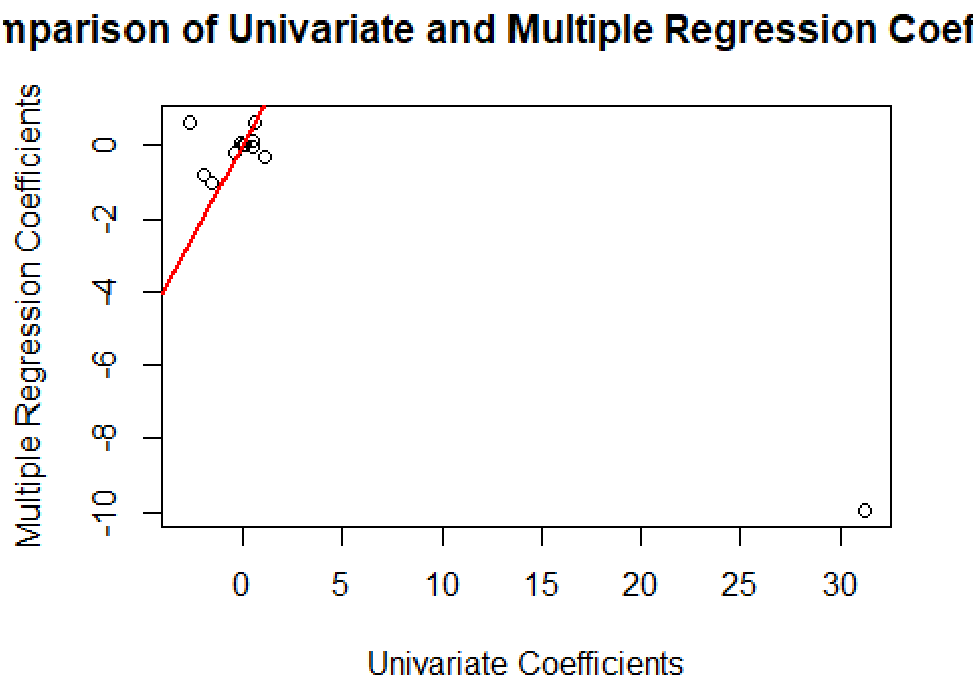
```r
#Fit simple linear regression models for each predictor and extract
coefficients
for (predictor in predictors) {
  formula <- as.formula(paste("crim ~", predictor))
  model <- lm(formula, data = Boston)
  univariate_coeffs <- c(univariate_coeffs, coef(model)[2])
}

#Display univariate coefficients
univariate_coeffs
```

```
##          zn       indus        chas         nox          rm         age
## -0.07393498  0.50977633 -1.89277655 31.24853120 -2.68405122  0.10778623
##         dis         rad         tax     ptratio       lstat        medv
## -1.55090168  0.61791093  0.02974225  1.15198279  0.54880478 -0.36315992
```

```r
#Fit a multiple regression model including all predictors
multi_model <- lm(crim ~ ., data = Boston)

#Extract coefficients for all predictors (excluding intercept)
multi_coeffs <- coef(multi_model)[-1]

#Display multiple regression coefficients
multi_coeffs
```

```
##            zn         indus          chas           nox            rm
##  0.0457100386 -0.0583501107 -0.8253775522 -9.9575865471  0.6289106622
##           age           dis           rad           tax       ptratio
## -0.0008482791 -1.0122467382  0.6124653115 -0.0037756465 -0.3040727572
##         lstat          medv
##  0.1388005968 -0.2200563590
```

```r
#Plot univariate vs multiple regression coefficients
plot(univariate_coeffs, multi_coeffs,
     xlab = "Univariate Coefficients",
     ylab = "Multiple Regression Coefficients",
     main = "Comparison of Univariate and Multiple Regression Coefficients")

#Add 45-degree reference line
abline(0, 1, col = "red", lwd = 2)
```




Comment: The plot shows that most predictors have similar coefficients in univariate and multiple regression models. However, "nox" (nitrogen oxides concentration (parts per 10 million)) is the only variable that exhibits a large discrepancy. This discrepancy likely results from multicollinearity with other predictors, as indicated by its extreme difference in coefficients.

## Part D

```
# List of all predictors in the dataset except for crim
predictors <- names(Boston)[names(Boston) != "crim"]

# Initialize a list to store significant cubic terms
significant_cubic <- list()

# Fit polynomial regression models for each predictor and check for cubic
terms
for (predictor in predictors) {
  formula <- as.formula(paste("crim ~", predictor, "+ I(", predictor, "^2) +
I(", predictor, "^3)"))
  poly_model <- lm(formula, data = Boston)
  coef_summary <- summary(poly_model)$coefficients

  # Check if the model includes the cubic term
  if (nrow(coef_summary) >= 4) {
    # Extract the p-value for the cubic term (4th row)
    p_value_cubic <- coef_summary[4, 4]
```

```
    if (p_value_cubic < 0.05) {
      significant_cubic <- c(significant_cubic, predictor)
    }
  }
}

# Display predictors with significant cubic terms
significant_cubic

## [[1]]
## [1] "indus"
##
## [[2]]
## [1] "nox"
##
## [[3]]
## [1] "age"
##
## [[4]]
## [1] "dis"
##
## [[5]]
## [1] "ptratio"
##
## [[6]]
## [1] "medv"
```

#"indus", "nox", "age", "dis", "ptratio", "medv" are all the variables that show a non-linear relationship with crime given the cubic polynomial model.

## Take-home #3 (Chapter 6 #11)

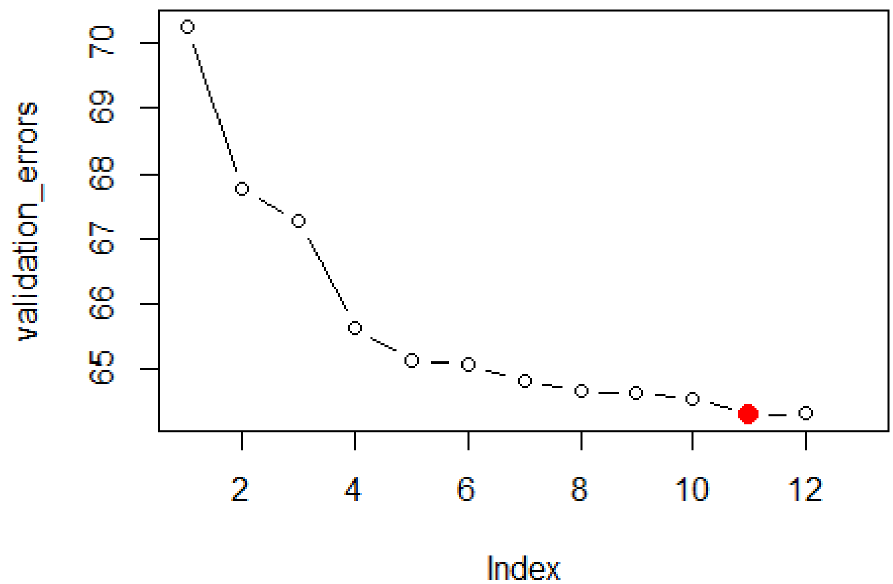
```
#Access Boston Data
library(ISLR2)
p3_data = Boston
summary(p3_data)

##       crim                zn             indus            chas
##  Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   Min.   :0.00000
##  1st Qu.: 0.08205   1st Qu.:  0.00   1st Qu.: 5.19   1st Qu.:0.00000
##  Median : 0.25651   Median :  0.00   Median : 9.69   Median :0.00000
##  Mean   : 3.61352   Mean   : 11.36   Mean   :11.14   Mean   :0.06917
##  3rd Qu.: 3.67708   3rd Qu.: 12.50   3rd Qu.:18.10   3rd Qu.:0.00000
##  Max.   :88.97620   Max.   :100.00   Max.   :27.74   Max.   :1.00000
##       nox               rm             age              dis
##  Min.   :0.3850   Min.   :3.561   Min.   :  2.90   Min.   : 1.130
##  1st Qu.:0.4490   1st Qu.:5.886   1st Qu.: 45.02   1st Qu.: 2.100
##  Median :0.5380   Median :6.208   Median : 77.50   Median : 3.207
```

```
##  Mean   :0.5547   Mean   :6.285   Mean   : 68.57   Mean   : 3.795  
##  3rd Qu.:0.6240   3rd Qu.:6.623   3rd Qu.: 94.08   3rd Qu.: 5.188  
##  Max.   :0.8710   Max.   :8.780   Max.   :100.00   Max.   :12.127  
##       rad              tax           ptratio          lstat      
##  Min.   : 1.000   Min.   :187.0   Min.   :12.60   Min.   : 1.73  
##  1st Qu.: 4.000   1st Qu.:279.0   1st Qu.:17.40   1st Qu.: 6.95  
##  Median : 5.000   Median :330.0   Median :19.05   Median :11.36  
##  Mean   : 9.549   Mean   :408.2   Mean   :18.46   Mean   :12.65  
##  3rd Qu.:24.000   3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:16.95  
##  Max.   :24.000   Max.   :711.0   Max.   :22.00   Max.   :37.97  
##       medv      
##  Min.   : 5.00  
##  1st Qu.:17.02  
##  Median :21.20  
##  Mean   :22.53  
##  3rd Qu.:25.00  
##  Max.   :50.00
```

## Part A

**Linear Model Regularization Method: Best Subset Selection**

```r
library(leaps)
set.seed(1)
train_set <- sample(1:nrow(p3_data), 0.80 * nrow(p3_data))
test_set <- setdiff(1:nrow(p3_data), train_set)
crim_test <- p3_data$crim[test_set]

#Perform best subset selection on training set
subset_fit <- regsubsets(crim ~ ., data = p3_data, subset = train_set, nvmax
= 13)
subset_summary <- summary(subset_fit)

#Create the test matrix
test_mat <- model.matrix(crim ~ ., data = p3_data[test_set, ])
validation_errors <- rep(NA, 13)

#Calculate validation errors for each model
for (idx in 1:12) {
    coefficients <- coef(subset_fit, id = idx)
    predictions <- test_mat[, names(coefficients)] %*% coefficients
    validation_errors[idx] <- mean((p3_data$crim[test_set] - predictions)^2)
}

#Identify the best model
optimal_model <- which.min(validation_errors)
plot(validation_errors, type = 'b')
points(optimal_model, validation_errors[optimal_model], col = "red", cex = 2,
pch = 20)
```

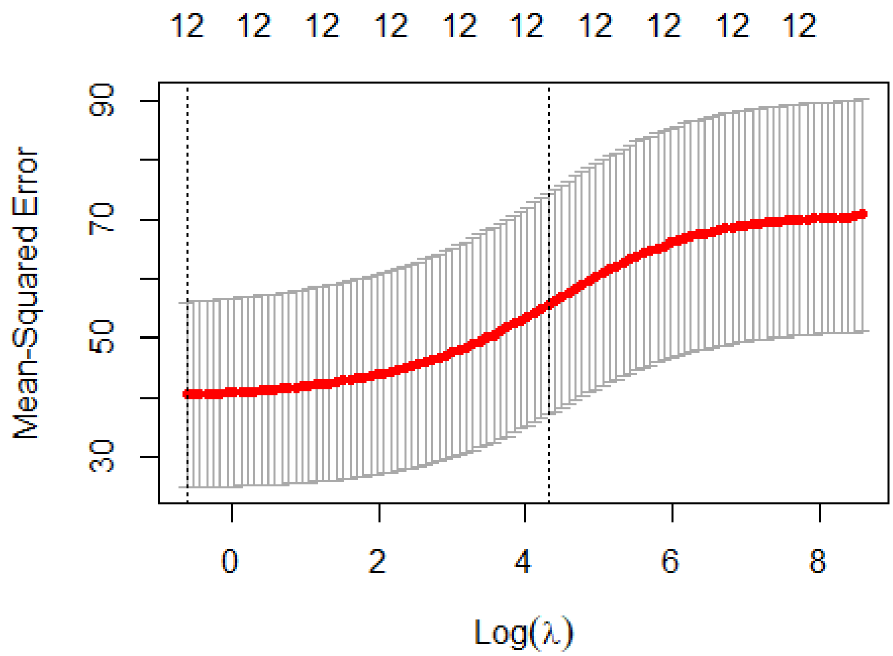



```
subset_mse <- validation_errors[optimal_model]
subset_rmse <- sqrt(subset_mse)

#Refit the best subset model on the entire dataset
full_subset_fit <- regsubsets(crim ~ ., data = p3_data, nvmax = 11)
full_model_coeffs <- coef(full_subset_fit, optimal_model)

#Display results
cat("Optimal number of predictors:", optimal_model, "\n")

## Optimal number of predictors: 11

cat("Coefficients of the best model:\n")

## Coefficients of the best model:

print(full_model_coeffs)

##   (Intercept)            zn         indus          chas           nox
##  13.801555544   0.045818028  -0.058345869  -0.828283847 -10.022404078
##            rm           dis           rad           tax       ptratio
##   0.623650191  -1.008539667   0.612822152  -0.003782956  -0.304784434
##         lstat          medv
##   0.137698956  -0.220092318

cat("MSE of the best model:", subset_mse, "\n")

## MSE of the best model: 64.2848
```

```r
cat("RMSE of the best model:", subset_rmse, "\n")

## RMSE of the best model: 8.01778
```

**Linear Model Regularization Method: Ridge Regression**

```r
library(glmnet)

## Loading required package: Matrix

## Loaded glmnet 4.1-8

#Set seed for reproducibility
set.seed(1)

#Prepare the data
X <- model.matrix(crim ~ ., p3_data)[, -1]  # Design matrix (features)
Y <- p3_data$crim  # Response vector (target variable)

# Split the data into training and testing sets
set.seed(4)
training_indices <- sample(1:nrow(p3_data), 0.80 * nrow(p3_data))
testing_indices <- setdiff(1:nrow(p3_data), training_indices)

#Fit the Ridge Regression model on the training set
ridge_model <- glmnet(X[training_indices, ], Y[training_indices], alpha = 0)

#Perform cross-validation to find the best lambda
cv_ridge_model <- cv.glmnet(X[training_indices, ], Y[training_indices], alpha
= 0)
plot(cv_ridge_model)
```




```
#Optimal lambda value
optimal_lambda <- cv_ridge_model$lambda.min
cat("Best lambda value for Ridge Regression:", optimal_lambda, "\n")

## Best lambda value for Ridge Regression: 0.5360017
```

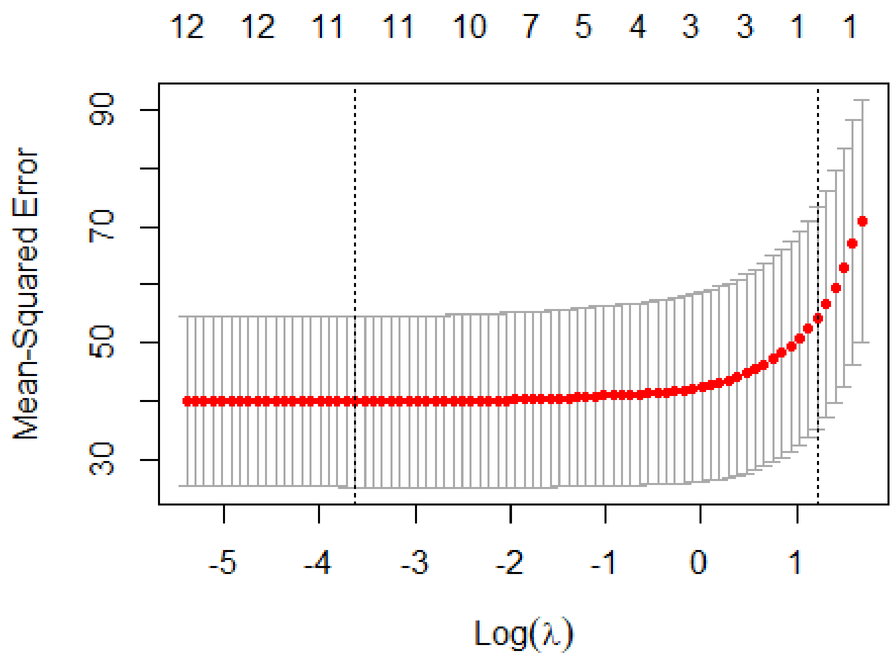
Note that I chose to do cross-validation to optimize hyper-parameter lamda.

```
#Predict on the test set using optimal lambda
ridge_predictions <- predict(ridge_model, s = optimal_lambda, newx =
X[testing_indices, ])
ridge_mse <- mean((ridge_predictions - Y[testing_indices])^2)
ridge_rmse <- sqrt(ridge_mse)

#Display the Ridge Regression Erro Values
cat("MSE for Ridge Regression:", ridge_mse, "\n")

## MSE for Ridge Regression: 51.27786

cat("RMSE for Ridge Regression:", ridge_rmse, "\n")

## RMSE for Ridge Regression: 7.160856
```

**Linear Model Regularization Method: Lasso Regression**

```
#Fit the Lasso Regression on training set
lasso_model <- glmnet(X[training_indices, ], Y[training_indices], alpha = 1)
```

```
#Perform cross-validation to find optimal lambda
cv_lasso_model <- cv.glmnet(X[training_indices, ], Y[training_indices], alpha
= 1)
plot(cv_lasso_model)
```




```
#Optimal lambda value
optimal_lambda <- cv_lasso_model$lambda.min
cat("Best lambda value for Lasso Regression:", optimal_lambda, "\n")

## Best lambda value for Lasso Regression: 0.02667693

#Predict on the test set using optimal lambda
lasso_predictions <- predict(lasso_model, s = optimal_lambda, newx =
X[testing_indices, ])
lasso_mse <- mean((lasso_predictions - Y[testing_indices])^2)
lasso_rmse <- sqrt(lasso_mse)

#Get coefficients of the best model
lasso_coefficients <- predict(lasso_model, type = "coefficients", s =
optimal_lambda)[1:13,]

#Results
cat("MSE for Lasso Regression:", lasso_mse, "\n")

## MSE for Lasso Regression: 50.83944

cat("RMSE for Lasso Regression:", lasso_rmse, "\n")
```

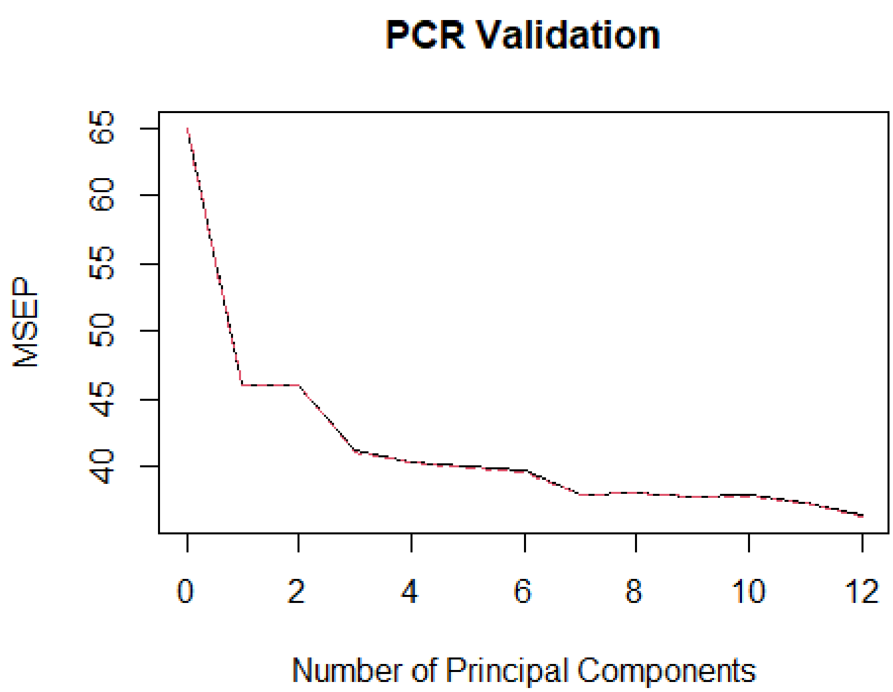
```
## RMSE for Lasso Regression: 7.130178

cat("Number of non-zero coefficients in the best model:",
length(lasso_coefficients[lasso_coefficients != 0]), "\n")

## Number of non-zero coefficients in the best model: 12

cat("Non-zero coefficients in the best model:\n")

## Non-zero coefficients in the best model:

print(lasso_coefficients[lasso_coefficients != 0])

##  (Intercept)          zn        indus         chas          nox
rm
## 12.179494455  0.041259779 -0.053172489 -0.400995114 -7.128511511
0.376115516
##          dis          rad          tax      ptratio        lstat
medv
## -0.886708399  0.574007848 -0.001985061 -0.242456123  0.076674319 -
0.206121819
```

**Linear Model Regularization Method: Principal Component Regression**

```
library(pls)

##
## Attaching package: 'pls'

## The following object is masked from 'package:stats':
##
##     loadings

#Fit the PCR model on the training set
pcr_model <- pcr(crim ~ ., data = p3_data, subset = train_set, scale = TRUE,
validation = "CV")
#Summary of PCR model
summary(pcr_model)

## Data:    X dimension: 404 12
##  Y dimension: 404 1
## Fit method: svdpc
## Number of components considered: 12
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           8.062    6.783    6.782    6.416    6.351    6.320    6.297
## adjCV        8.062    6.781    6.779    6.411    6.344    6.317    6.293
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps
## CV       6.162    6.167    6.145     6.151     6.111     6.029
## adjCV    6.157    6.163    6.140     6.146     6.105     6.022
```

```
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8
comps
## X      50.37    64.05    73.18    80.30    86.59    90.14    92.71
94.86
## crim   29.85    30.02    37.77    38.96    39.64    40.17    42.82
43.05
##       9 comps  10 comps  11 comps  12 comps
## X      96.71     98.23     99.44    100.00
## crim   43.45     43.53     44.50     46.03
```

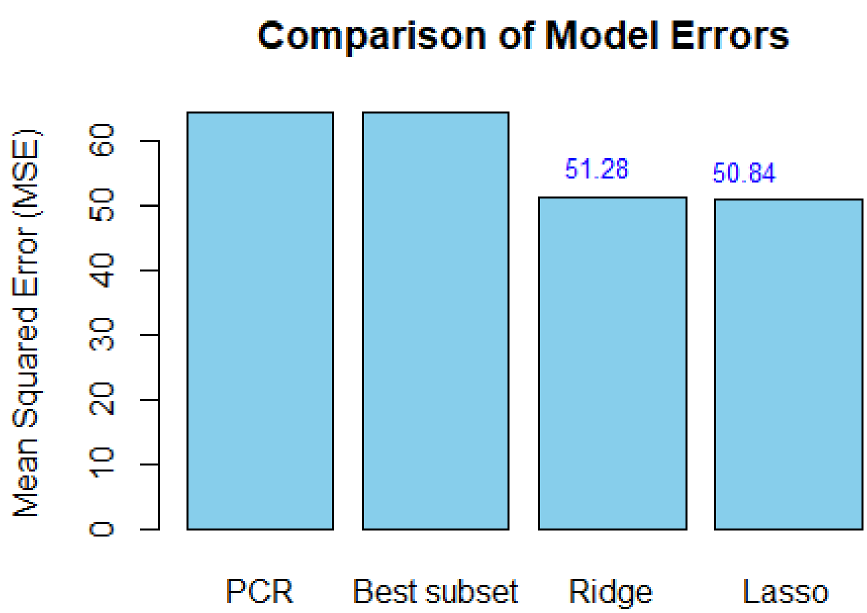
```
validationplot(pcr_model, val.type = "MSEP", main = "PCR Validation", xlab =
"Number of Principal Components")
```




```
#Make predictions on the test set using the model with 12 components
pcr_predictions <- predict(pcr_model, p3_data[test_set,], ncomp = 12)
pcr_mse <- mean((pcr_predictions - p3_data$crim[test_set])^2)
pcr_rmse <- sqrt(pcr_mse)

#Display the PCR results
cat("MSE for PCR:", pcr_mse, "\n")

## MSE for PCR: 64.30352

cat("RMSE for PCR:", pcr_rmse, "\n")

## RMSE for PCR: 8.018948
```

### Part B

```
#Combine the errors into a vector
errors <- c(subset_mse, ridge_mse, lasso_mse, pcr_mse)
names(errors) <- c("Best subset", "Ridge", "Lasso", "PCR")

#Plot the errors
barplot(sort(errors, decreasing = TRUE), col = "skyblue",
        main = "Comparison of Model Errors", ylab = "Mean Squared Error
(MSE)")

text(x = seq_along(errors), y = sort(errors, decreasing = TRUE),
     labels = round(sort(errors, decreasing = TRUE), 2), pos = 3, cex = 0.8,
col = "blue")
```




Summary of Results Among the 4 models Specified:

```
MSE of the best model: 64.2848  RMSE of the best model: 8.01778

MSE for Ridge Regression: 51.27786  RMSE for Ridge Regression: 7.160856

MSE for Lasso Regression: 50.83944  RMSE for Lasso Regression: 7.130178

MSE for PCR: 64.30352  RMSE for PCR: 8.018948
```

Among the four models considered, Lasso Regression yielded the lowest RMSE, making it the best-performing model for predicting the per capita crime rate in the Boston dataset. The lasso model demonstrated that it can perform variable selection and regularization.

The lower RMSE as compared to the other models shows that it can better manage multicollinearity and prevent overfitting, leading to improved predictive accuracy. Additionaly, this particular model used cross-validation to optimise the hyper-parameter lambda which helped reduce overfitting and improved the robustness of the model. It is to note that Ridge was not very far off and used a similar cross-validation method.

### Part C

The Lasso Regression model selected 11 features and the intercept term, excluding age. This may have reduced the complexity of the model to improve interpretability and predictive accuracy by focusing on the more important variables.

## Take-home #4 (Chapter 8 #8)

```r
# Load necessary libraries
library(readr)
library(dplyr)
library(tree)
library(rpart)
library(rpart.plot)
library(randomForest)
```

```
## randomForest 4.7-1.1

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin

## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
library(caret)
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:pls':
##
##     R2
```

```r
library(BART)
```

```
## Loading required package: nlme
```

```
##
## Attaching package: 'nlme'

## The following object is masked from 'package:dplyr':
##
##     collapse

## Loading required package: survival

##
## Attaching package: 'survival'

## The following object is masked from 'package:caret':
##
##     cluster

# Load the dataset
#Load in the Austin Housing Data
austin_data <- read_csv('austinhouses.csv',rt)

## Rows: 6785 Columns: 34

## ── Column specification
────────────────────────────────────────────────────────

## Delimiter: ","
## chr (34): X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14,
X15, ...
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this
message.

#Set the first row as column names
colnames(austin_data) <- austin_data[1, ]
austin_data <- austin_data[-1, ]
rownames(austin_data) <- NULL


# Convert relevant columns to numeric and handle non-numeric values
austin_data <- austin_data %>%
  mutate(
    latestPrice = as.numeric(latestPrice),
    latitude = as.numeric(latitude),
    longitude = as.numeric(longitude),
    hasAssociation = as.factor(hasAssociation), # Assuming this is a
categorical variable
    livingAreaSqFt = as.numeric(livingAreaSqFt),
    numOfBathrooms = as.numeric(numOfBathrooms),
    numOfBedrooms = as.numeric(numOfBedrooms)
  )
```

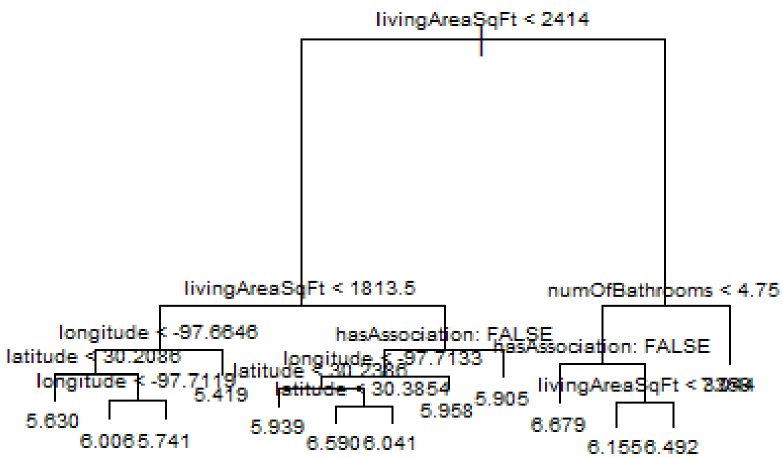
```r
# Check for and handle any NA values
austin_data <- austin_data %>%
  filter(
    !is.na(latestPrice) &
    !is.na(latitude) &
    !is.na(longitude) &
    !is.na(hasAssociation) &
    !is.na(livingAreaSqFt) &
    !is.na(numOfBathrooms) &
    !is.na(numOfBedrooms)
  )

# Add column for log(latestPrice)
austin_data <- austin_data %>%
  mutate(log_latestPrice = log(latestPrice))

# Initialize a data frame to store MSE and RMSE values
results <- data.frame(
  Model = character(),
  MSE = numeric(),
  RMSE = numeric(),
  stringsAsFactors = FALSE
)
```

## Part A

```r
#Split the data
set.seed(1)
train <- sample(1:nrow(austin_data), 0.8*nrow(austin_data))
austin_data.train <- austin_data[train, ]
austin_data.test <- austin_data[-train, ]
```

## Part B

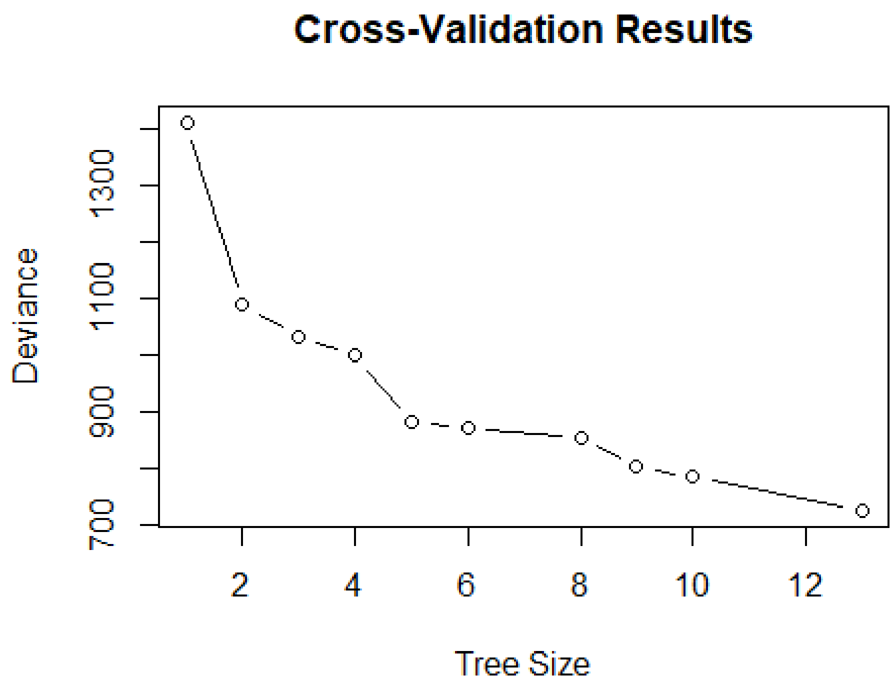
```r
#Fit regression tree
tree.austin <- tree(log_latestPrice ~ latitude + longitude + hasAssociation +
livingAreaSqFt + numOfBathrooms + numOfBedrooms,
                    data = austin_data.train)

#Plot
plot(tree.austin)
text(tree.austin, pretty = 0, cex = 0.6, xpd = NA, offset = 0.5)
```



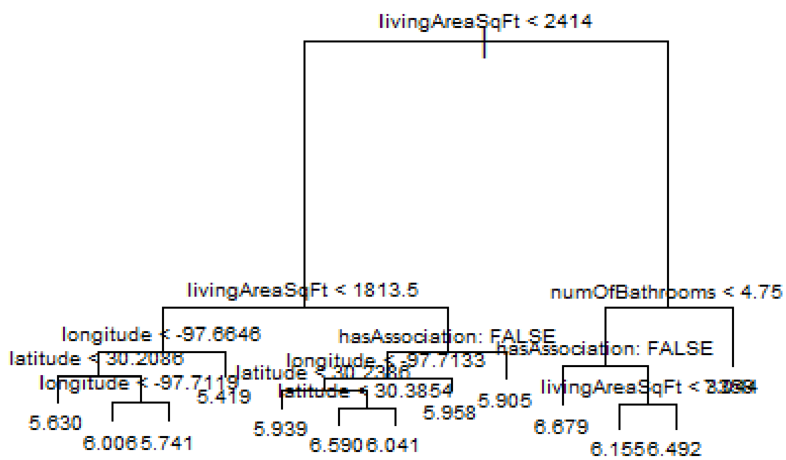

```r
#Predict on test set
predictions <- predict(tree.austin, newdata = austin_data.test)

#Convert predictions back to price scale
predicted_prices <- exp(predictions)
actual_prices <- austin_data.test$latestPrice

#Calculate MSE and RMSE
tree_mse <- mean((predicted_prices - actual_prices)^2)
tree_rmse <- sqrt(tree_mse)

cat("Test MSE: ", tree_mse, "\n")

## Test MSE:  71405.56

cat("Test RMSE: ", tree_rmse, "\n")

## Test RMSE:  267.2182
```

## Part C

```r
#Perform cross-validation to determine the optimal level of tree complexity
cv.austin <- cv.tree(tree.austin, FUN = prune.tree)

#Plot cross-validation results
plot(cv.austin$size, cv.austin$dev, type = "b", xlab = "Tree Size", ylab =
"Deviance", main = "Cross-Validation Results")
```




```r
#Find the optimal size for pruning
optimal_size <- cv.austin$size[which.min(cv.austin$dev)]

#Prune the tree to the optimal size
pruned.austin <- prune.tree(tree.austin, best = optimal_size)

#Plot the pruned tree
plot(pruned.austin)
text(pruned.austin, pretty = 0, cex = 0.6, xpd = NA, offset = 0.5)
```

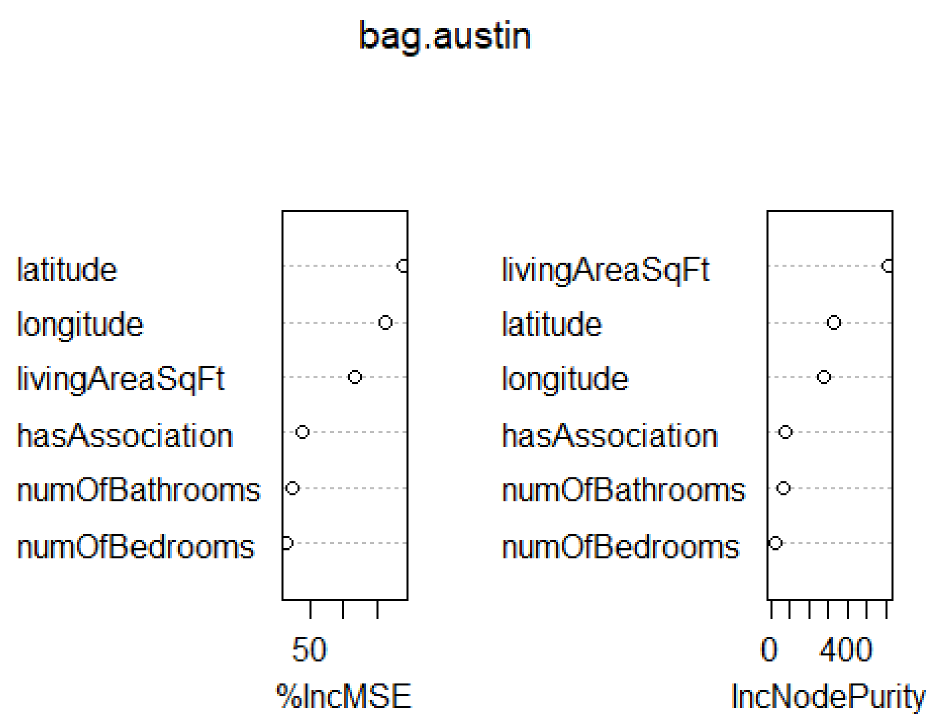



```r
#MSE and RMSE for Tree before CV and Pruning
mse_original <- mean((predicted_prices - actual_prices)^2)
rmse_original <- sqrt(mse_original)

cat("Original Tree Test MSE: ", tree_mse, "\n")

## Original Tree Test MSE:  71405.56

cat("Original Tree Test RMSE: ", tree_rmse, "\n")

## Original Tree Test RMSE:  267.2182

#Predict on test set using the pruned tree
pruned_yhat <- predict(pruned.austin, newdata = austin_data.test)
pruned_predicted_prices <- exp(pruned_yhat) # Convert back to price scale

#Calculate MSE and RMSE for the pruned tree
mse_pruned <- mean((pruned_predicted_prices - actual_prices)^2)
rmse_pruned <- sqrt(mse_pruned)

cat("Pruned Tree Test MSE: ", mse_pruned, "\n")

## Pruned Tree Test MSE:  71405.56

cat("Pruned Tree Test RMSE: ", rmse_pruned, "\n")

## Pruned Tree Test RMSE:  267.2182
```

#Pruning does not improve the TEST MSE. It remains the same.

## Part D

```
library(randomForest)

set.seed(1)
bag.austin <- randomForest(log_latestPrice ~ latitude + longitude +
hasAssociation + livingAreaSqFt + numOfBathrooms + numOfBedrooms,
                           data = austin_data.train,
                           mtry = 6, # Number of predictors
                           importance = TRUE)

#Predict on test set using the bagging model
yhat.bag <- predict(bag.austin, newdata = austin_data.test)
predicted_prices_bag <- exp(yhat.bag) # Convert back to price scale
actual_prices <- austin_data.test$latestPrice
#Calculate MSE and RMSE for the bagging model
mse_bag <- mean((predicted_prices_bag - actual_prices)^2)
rmse_bag <- sqrt(mse_bag)

#Determine the importance of each variable
importance_values <- importance(bag.austin)
varImpPlot(bag.austin)
```




Comments:

1. **Latitude and Longitude**: - Both `latitude` and `longitude` have high %IncMSE and IncNodePurity values, indicating that they are very important predictors in the model. The geographical location of a property significantly impacts its price.

2. **Living Area (SqFt)**:
   - `livingAreaSqFt` has the highest IncNodePurity, showing its substantial contribution to reducing node impurity. It's also quite important in terms of %IncMSE, suggesting that larger living areas are closely associated with higher property prices.

### Part E

```
#Fit random forest model
set.seed(1)
rf.austin <- randomForest(log_latestPrice ~ latitude + longitude +
hasAssociation + livingAreaSqFt + numOfBathrooms + numOfBedrooms,
                          data = austin_data.train,
                          mtry = 3,
                          importance = TRUE)
#Summary
print(rf.austin)

##
## Call:
##  randomForest(formula = log_latestPrice ~ latitude + longitude +
hasAssociation + livingAreaSqFt + numOfBathrooms + numOfBedrooms,      data =
austin_data.train, mtry = 3, importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##           Mean of squared residuals: 0.07695819
##                     % Var explained: 70.39

#Predict on test set using the random forest model
yhat.rf <- predict(rf.austin, newdata = austin_data.test)
predicted_prices_rf <- exp(yhat.rf) # Convert back to price scale
actual_prices <- austin_data.test$latestPrice

#Calculate MSE and RMSE for the random forest model
mse_rf <- mean((predicted_prices_rf - actual_prices)^2)
rmse_rf <- sqrt(mse_rf)


cat("Random Forest Model Test MSE: ", mse_rf, "\n")

## Random Forest Model Test MSE:  40398.32

cat("Random Forest Model Test RMSE: ", rmse_rf, "\n")

## Random Forest Model Test RMSE:  200.9933
```

```r
#Determine the importance of each variable
importance_values_rf <- importance(rf.austin)
print(importance_values_rf)
```

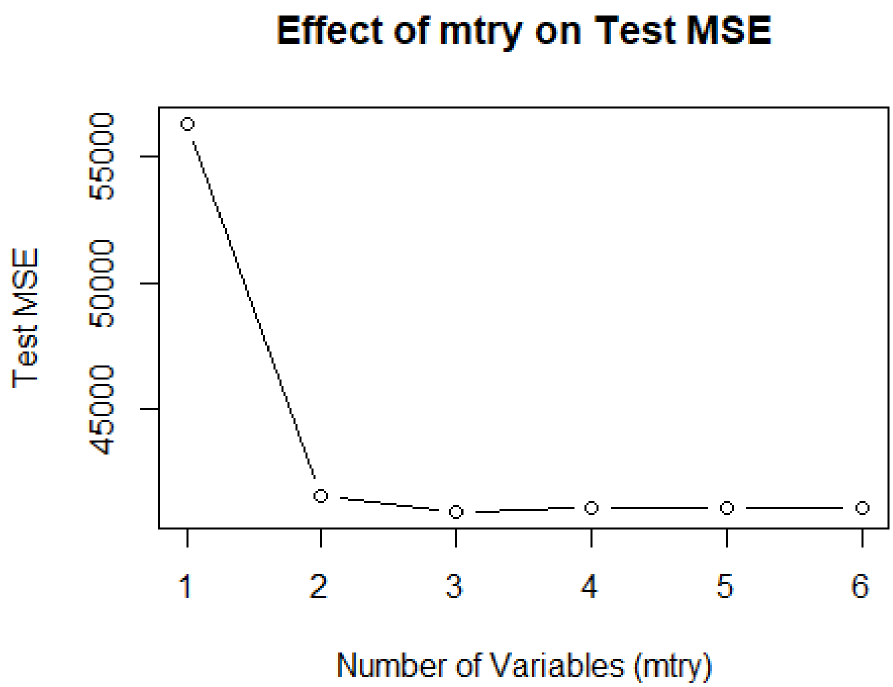
```
##                    %IncMSE IncNodePurity
## latitude        165.12645     332.67981
## longitude       139.50568     286.19302
## hasAssociation   36.09719      61.72757
## livingAreaSqFt   69.85699     462.43869
## numOfBathrooms   28.80979     162.82788
## numOfBedrooms    17.06917      50.41653
```

```r
#Plot variable importance
varImpPlot(rf.austin)
```


# Comments: Geographical location (latitude and longitude) and the living area are the most significant predictors of property prices in Austin. The presence of a homeowners' association and the number of bathrooms also contribute but to a lesser extent. The number of bedrooms is the least important predictor among the variables considered.

```r
#Test different values of mtry
set.seed(1)
mtry_values <- c(1, 2, 3, 4, 5, 6)
mse_values <- c()

for (m in mtry_values) {
  rf_model <- randomForest(log_latestPrice ~ latitude + longitude +
```

```
hasAssociation + livingAreaSqFt + numOfBathrooms + numOfBedrooms,
                           data = austin_data.train,
                           mtry = m,
                           importance = TRUE)
  yhat <- predict(rf_model, newdata = austin_data.test)
  predicted_prices <- exp(yhat)
  mse <- mean((predicted_prices - actual_prices)^2)
  mse_values <- c(mse_values, mse)
}

#Plot the effect of mtry on MSE
plot(mtry_values, mse_values, type = "b", xlab = "Number of Variables
(mtry)", ylab = "Test MSE", main = "Effect of mtry on Test MSE")
```




Seeing the plot and comparing the RMSE's for mytry = 6 and 3, at mtry = 3, we find the lowest RMSE. This means that the model performed better with a smaller number but only specifically at mytry = 3 of features selected at each node which is interesting. This can improve model accuracy.

mytry = 6

```
Random Forest Model Test MSE:  41019.49  Random Forest Model Test RMSE:
202.5327
```

mytry = 3

```
Random Forest Model Test MSE:  40398.32  Random Forest Model Test RMSE:
200.9933
```

## Part F

```
#Preparing Data for BART
#Install Bart Package
if (!requireNamespace("BART", quietly = TRUE)) {
  install.packages("BART")
}

#Load Libraries
library(readr)
library(dplyr)
library(BART)

#Load Data
austin_data <- read_csv('austinhouses.csv', rt)
```

```
## Rows: 6785 Columns: 34
## ── Column specification
────────────────────────────────────────────────────────
## Delimiter: ","
## chr (34): X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14,
X15, ...
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this
message.
```

```
#Data set Cleaning
colnames(austin_data) <- austin_data[1, ]
austin_data <- austin_data[-1, ]
rownames(austin_data) <- NULL

austin_data <- austin_data %>%
  mutate(
    latestPrice = as.numeric(latestPrice),
    latitude = as.numeric(latitude),
    longitude = as.numeric(longitude),
    hasAssociation = as.numeric(as.factor(hasAssociation)), # Convert factor
to numeric
    livingAreaSqFt = as.numeric(livingAreaSqFt),
    numOfBathrooms = as.numeric(numOfBathrooms),
    numOfBedrooms = as.numeric(numOfBedrooms)
  )

austin_data <- austin_data %>%
  filter(
    !is.na(latestPrice) &
    !is.na(latitude) &
```

```
    !is.na(longitude) &
    !is.na(hasAssociation) &
    !is.na(livingAreaSqFt) &
    !is.na(numOfBathrooms) &
    !is.na(numOfBedrooms)
  )

# Add column for log(latestPrice)
austin_data <- austin_data %>%
  mutate(log_latestPrice = log(latestPrice))

# Split the data according to 80/20 split (80 train 20 test) + set seed to 1
to ensure reproducibility
set.seed(1)
train <- sample(1:nrow(austin_data), 0.8 * nrow(austin_data))
austin_data.train <- austin_data[train, ]
austin_data.test <- austin_data[-train, ]

# Prepare the data for BART
xtrain <- austin_data.train %>% select(latitude, longitude, hasAssociation,
livingAreaSqFt, numOfBathrooms, numOfBedrooms)
ytrain <- austin_data.train$log_latestPrice
xtest <- austin_data.test %>% select(latitude, longitude, hasAssociation,
livingAreaSqFt, numOfBathrooms, numOfBedrooms)
ytest <- austin_data.test$log_latestPrice

# Ensure the data is in the correct format
xtrain <- as.data.frame(lapply(xtrain, as.numeric))
xtest <- as.data.frame(lapply(xtest, as.numeric))

# Fit the BART model
set.seed(1)
bartfit <- gbart(xtrain, ytrain, x.test = xtest)

## *****Calling gbart: type=1
## *****Data:
## data:n,p,np: 5427, 6, 1357
## y1,yn: 0.455933, -0.105536
## x1,x[n*p]: 30.341196, 2.000000
## xp1,xp[np*p]: 30.488775, 3.000000
## *****Number of Trees: 200
## *****Number of Cut Points: 100 ... 8
## *****burn,nd,thin: 100,1000,1
## *****Prior:beta,alpha,tau,nu,lambda,offset:
2,0.95,0.130101,3,0.027687,6.05878
## *****sigma: 0.377010
## *****w (weights): 1.000000 ... 1.000000
## *****Dirichlet:sparse,theta,omega,a,b,rho,augment: 0,0,1,0.5,1,6,0
## *****printevery: 100
##
## MCMC
```

```
## done 0 (out of 1100)
## done 100 (out of 1100)
## done 200 (out of 1100)
## done 300 (out of 1100)
## done 400 (out of 1100)
## done 500 (out of 1100)
## done 600 (out of 1100)
## done 700 (out of 1100)
## done 800 (out of 1100)
## done 900 (out of 1100)
## done 1000 (out of 1100)
## time: 61s
## trcnt,tecnt: 1000,1000
```

```
# Print the model summary
summary(bartfit)
```

```
##                  Length  Class     Mode
## sigma               1100 -none-    numeric
## yhat.train       5427000 -none-    numeric
## yhat.test        1357000 -none-    numeric
## varcount            6000 -none-    numeric
## varprob             6000 -none-    numeric
## treedraws              2 -none-    list
## proc.time              5 proc_time numeric
## hostname               1 -none-    logical
## yhat.train.mean     5427 -none-    numeric
## sigma.mean             1 -none-    numeric
## LPML                   1 -none-    numeric
## yhat.test.mean      1357 -none-    numeric
## ndpost                 1 -none-    numeric
## offset                 1 -none-    numeric
## varcount.mean          6 -none-    numeric
## varprob.mean           6 -none-    numeric
## rm.const               6 -none-    numeric
```

```
#Predict on test set using BART
yhat.bart <- bartfit$yhat.test.mean
predicted_prices_bart <- exp(yhat.bart) # Convert back to price scale
actual_prices <- exp(ytest) # Convert back to price scale

#MSE and RMSE
mse_bart <- mean((predicted_prices_bart - actual_prices)^2)
rmse_bart <- sqrt(mse_bart)


cat("BART Model Test MSE: ", mse_bart, "\n")
```

```
## BART Model Test MSE:  42308.27
```

```
cat("BART Model Test RMSE: ", rmse_bart, "\n")
```

```
## BART Model Test RMSE:  205.6897
```

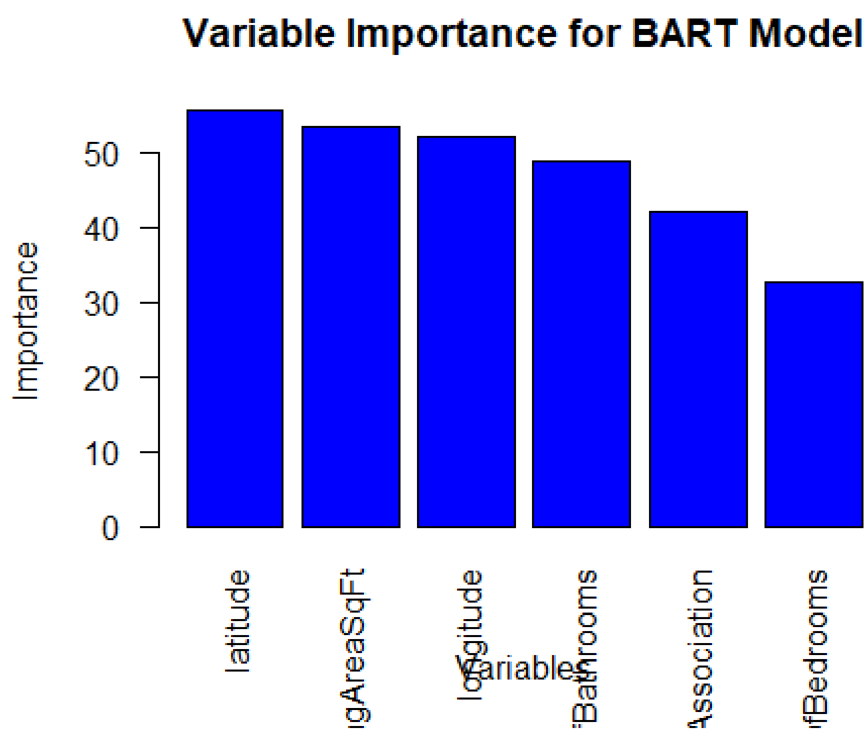
```r
#Importance
ord <- order(bartfit$varcount.mean, decreasing = TRUE)
variable_importance <- bartfit$varcount.mean[ord]

#Print Statement
cat("Variable Importance (sorted): \n")
```

```
## Variable Importance (sorted):
```

```r
print(variable_importance)
```

```
##       latitude livingAreaSqFt      longitude numOfBathrooms hasAssociation
##         55.591         53.406         52.052         48.882         41.989
##   numOfBedrooms
##         32.624
```

```r
# Plot variable importance
barplot(variable_importance, main = "Variable Importance for BART Model",
        xlab = "Variables", ylab = "Importance",
        names.arg = names(variable_importance), las = 2, col = "blue")
```



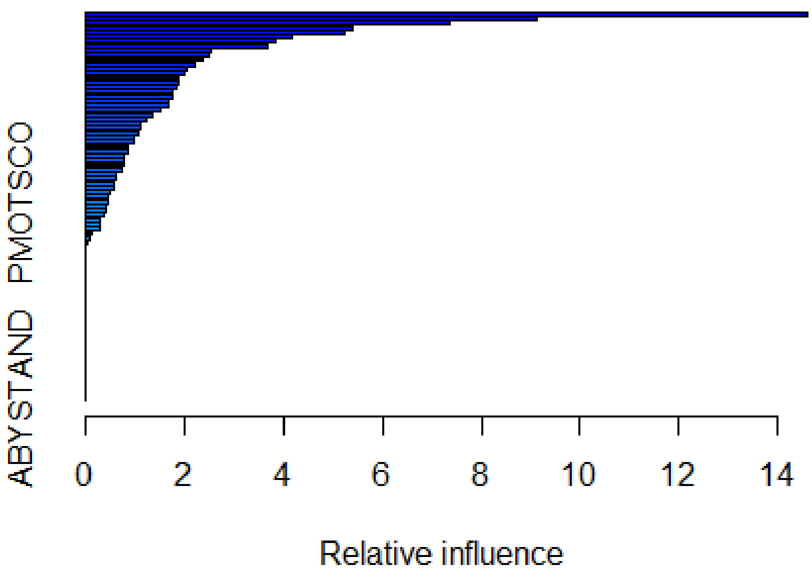
## Take-home #5 (Chapter 8 #8)

### Part A

```
#Load Libraries
library(ISLR)

##
## Attaching package: 'ISLR'

## The following objects are masked from 'package:ISLR2':
##
##     Auto, Credit

library(dplyr)
p5_data <- Caravan
#Data Cleaning and Adjustments
p5_data <- p5_data %>%
  mutate(Purchase = ifelse(Purchase == "Yes", 1, 0))
# Create a training set consisting of the first 1,000 observations
p5_train <- p5_data %>% slice(1:1000)
# Create a test set consisting of the remaining observations
p5_test <- p5_data %>% slice(1001:n())
```

### Part B

```
library('gbm')

## Loaded gbm 2.2.2

## This version of gbm is no longer under development. Consider transitioning
to gbm3, https://github.com/gbm-developers/gbm3

boost_model <- gbm(Purchase ~ ., data = p5_train, distribution = "bernoulli",
n.trees = 1000, shrinkage = 0.01)

## Warning in gbm.fit(x = x, y = y, offset = offset, distribution =
distribution,
## : variable 50: PVRAAUT has no variation.

## Warning in gbm.fit(x = x, y = y, offset = offset, distribution =
distribution,
## : variable 71: AVRAAUT has no variation.

summary(boost_model)[1:10, ]
```




```
##                   var   rel.inf
## PPERSAUT PPERSAUT 14.608460
## MKOOPKLA MKOOPKLA  9.124212
## MOPLHOOG MOPLHOOG  7.357535
## MBERMIDD MBERMIDD  5.409960
## PBRAND      PBRAND  5.239352
## MGODGE      MGODGE  4.192046
## ABRAND      ABRAND  3.850228
## MINK3045 MINK3045  3.704873
## MOSTYPE    MOSTYPE  2.536196
## PWAPART    PWAPART  2.514622
```

Comment: The table shows the top 10 predictors with the highest relative influence of different for predicting the Purchase. PPERSAUT is at the top with a 14.48 relative influence.

## Part C

```
#Predict the response on test data
predicted_probabilities <- predict(boost_model, p5_test, n.trees = 1000, type
= "response")
predicted_labels <- ifelse(predicted_probabilities > 0.2, 1, 0)

#Confusion matrix
confusion_matrix_boost <- table(p5_test$Purchase, predicted_labels)
#Calculate precision for boosting model
precision_boost <- confusion_matrix_boost[2, 2] /
```

```r
sum(confusion_matrix_boost[, 2])
#Calculate sensitivity for boosting model
sensitivity_boost <- confusion_matrix_boost[2, 2] /
sum(confusion_matrix_boost[2, ])
print(confusion_matrix_boost)
```

```
##    predicted_labels
##        0    1
##   0 4421  112
##   1  257   32
```

```r
cat("Boosting Model Precision:", precision_boost, "\n")
```

```
## Boosting Model Precision: 0.2222222
```

```r
cat("Boosting Model Sensitivity:", sensitivity_boost, "\n")
```

```
## Boosting Model Sensitivity: 0.1107266
```

```r
#Fit logistic regression model for comparison
logistic_model <- glm(Purchase ~ ., data = p5_train, family = binomial)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
#Predict response on test data using logistic regression
logistic_probabilities <- predict(logistic_model, p5_test, type = "response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type
== :
## prediction from rank-deficient fit; attr(*, "non-estim") has doubtful
cases
```

```r
logistic_labels <- ifelse(logistic_probabilities > 0.2, 1, 0)

#Create confusion matrix for logistic regression
confusion_matrix_logistic <- table(p5_test$Purchase, logistic_labels)

#Calculate precision for logistic regression
precision_logistic <- confusion_matrix_logistic[2, 2] /
sum(confusion_matrix_logistic[, 2])

#Calculate sensitivity for logistic regression
sensitivity_logistic <- confusion_matrix_logistic[2, 2] /
sum(confusion_matrix_logistic[2, ])

print(confusion_matrix_logistic)
```

```
##    logistic_labels
##        0    1
##   0 4183  350
##   1  231   58
```

```r
cat("Logistic Regression Precision:", precision_logistic, "\n")
```

```
## Logistic Regression Precision: 0.1421569
```

```r
cat("Logistic Regression Sensitivity:", sensitivity_logistic, "\n")
```

```
## Logistic Regression Sensitivity: 0.200692
```

Comments:

Boosting model has higher precision (21.12%) and lower sensitivity (11.76%) compared to logistic regression with lower precision (14.22%) and higher sensitivity (20.07%). Boosting model is more accurate when it predicts a purchase, but the logistic regression model is better at identifying actual purchasers. There is a trade off in this case because the choice between models depends on whether minimizing false positives (boosting) or maximizing the detection of actual purchasers where logistic regression may apply better.

## Group Project Contribution:

In the group project, I was responsible for building the KNN classification model. Aside from this I also did a little bit of exploratory analysis to find out what variables were correlated with each other – I helped with the heat map. Besides programming, I was present in the meetings and maintained communication over slack and the groupchat.