Problem set 1

Tongxin Zhu

1. Canonical Data Mining Tasks
   a. This scenario is a **regression** problem since this problem is about predicting a continuous quantity output, CEO salary. The number of observations **n** is **500** since the number of firms that we collected data is 500, and the number of predictions **p** is **3** since for each firm we record profit, number of employees, and industry which is 3 predictions in total.
   b. This scenario is a **classification** problem since this problem is about predicting a discrete class label output, whether a product will be a success or a failure. The number of observations **n** is **20** since the number of past products that we collected data is 20, and the number of predictions **p** is **13** since for each product we record the price charge, marketing budget, competition price, and ten other variables which is 13 predictions in total.
   c. This scenario is a **regression** problem since this problem is about predicting a continuous quantity output, the % change in US dollar. The number of observations **n** is **52** since we collected weekly data for all of 2020 that has 52 weeks, and the number of predictions **p** is **3** since for each week we record the % change in the US market, the % change in the British market, and the % change in the German market which is 3 predictions in total.

2. Linear Regression - Inference (p-Values)
   a. TV vs Sales
      The hypothesis of linear regression:
      - H_0: There is no linear relationship between the amount invested in TV advertising and sales
      - H_1: There is linear relationship between the amount invested in TV advertising and sales
      - The P-value corresponding to TV advertising is less than 0.0001 which is less than the significance level, 0.05, hence the null hypothesis H_0 should be rejected which means **there is linear relationship between the amount invested in TV advertising and sales**
   b. Radio vs Sales
      - H_0: There is no linear relationship between the amount invested in radio advertising and sales
      - H_1: There is linear relationship between the amount invested in radio advertising and sales
      - The P-value corresponding to radio advertising is less than 0.0001 which is less than the significance level, 0.05, hence the null hypothesis H_0 should be rejected which means **there is linear relationship between the amount invested in radio advertising and sales**

c. Newspaper vs Sales
   - H_0: There is no linear relationship between the amount invested in newspaper advertising and sales
   - H_1: There is linear relationship between the amount invested in newspaper advertising and sales
   - The P-value corresponding to newspaper advertising is equal to 0.8599 which is higher than the significance level, 0.05, hence the hypothesis H_1 should be rejected which means **there is no linear relationship between the amount invested in newspaper advertising and sales**

3. Linear Regression - Estimation (Coefficients)

a. Based on the information provided, we can get the following equation:
   Salary = 50 + 20*GPA + 0.07*IQ + 35*Female + 0.01*GPA*IQ - 10*GPA*Female
   1. Males salary = 50 + 20*GPA + 0.07*IQ + 0.01*GPA*IQ
      Female salary = 50 + 20*GPA + 0.07*IQ + 35 + 0.01*GPA*IQ - 10*GPA
      By subtracting out the common terms, females can earn 35 - 10*GPA more than male. Since we do not know the value of fixed GPA, we cannot know if males can earn more on average than females, so this statement is incorrect.
   2. Same as above, this statement is incorrect.
   3. Based on the conclusion of statement 1, we know that females can earn 35 - 10*GPA more than male with fixed GPA and IQ which means if GPA is high enough, females would earn less. Thus, this statement is correct.
   4. Same as above, this statement is incorrect.
      Thus, the answer is statement 3.

b. Based on the formula from question a, we can calculate the salary of a female by: 50 + 20*4 + 0.07*110 + 35 + 0.01*4*110 - 10*4 = 137.1
   Since the unit provided is in thousands of dollars, the salary of a female should be **$137100**

c. **False.** The statistical significance and interaction effect of a term does not depend on how large the coefficient is.

4. Classification

I think the reason this kind of misclassification happened might be that our prediction model is overfitting. Overfitting refers to the condition when the model completely fits the training data but fails to generalize the testing unseen data. In the classification trees, overfitting occurs when the tree is designed super perfect that can iti all samples in the training data; however, it is too perfect thus it ends up with too strict rules to classify data, which will lead to the low accuracy when predicting samples in testing data. Under this situation, a classification tree model might produce a split with two terminal nodes with the same label. Tree induction commonly uses two techniques to avoid overfitting which are: stop growing the tree before it gets too complex and grow the tree until it is too large, then prune it back. For example, we can specify a minimum number of instances that must be present in a leaf to limit the tree size, or if the model is done already, we can trim off some branches of the tree. We can use the cross-validation method to find the best model with the highest accuracy.

Please read R comments in the screen shots for following questions

5. Vectors and Computations

```
# 5
# a
result <- 0
for (i in 10:25){
  tem = i^4 + i^5
  result = result + tem
}
print(result)
# b
tem <- 10:25
print(sum(tem^4 + tem^5))
```

```
> result <- 0
> for (i in 10:25){
+   tem = i^4 + i^5
+   result = result + tem
+ }
> print(result)
[1] 47753112
> tem <- 10:25
> print(sum(tem^4 + tem^5))
[1] 47753112
```

6. Working with Character Vectors

```
# 6
# a
paste("label", 1:30, sep = " ")
# b
paste("fn", 1:30, sep = "")
sprintf("fn%d", 1:30)
```

```
> paste("label", 1:30, sep = " ")
 [1] "label 1"  "label 2"  "label 3"  "label 4"  "label 5"  "label 6"  "label 7"  "label 8"
 [9] "label 9"  "label 10" "label 11" "label 12" "label 13" "label 14" "label 15" "label 16"
[17] "label 17" "label 18" "label 19" "label 20" "label 21" "label 22" "label 23" "label 24"
[25] "label 25" "label 26" "label 27" "label 28" "label 29" "label 30"
> paste("fn", 1:30, sep = "")
 [1] "fn1"  "fn2"  "fn3"  "fn4"  "fn5"  "fn6"  "fn7"  "fn8"  "fn9"  "fn10" "fn11" "fn12" "fn13"
[14] "fn14" "fn15" "fn16" "fn17" "fn18" "fn19" "fn20" "fn21" "fn22" "fn23" "fn24" "fn25" "fn26"
[27] "fn27" "fn28" "fn29" "fn30"
> sprintf("fn%d", 1:30)
 [1] "fn1"  "fn2"  "fn3"  "fn4"  "fn5"  "fn6"  "fn7"  "fn8"  "fn9"  "fn10" "fn11" "fn12" "fn13"
[14] "fn14" "fn15" "fn16" "fn17" "fn18" "fn19" "fn20" "fn21" "fn22" "fn23" "fn24" "fn25" "fn26"
[27] "fn27" "fn28" "fn29" "fn30"
```

7. Understanding Vectorized Instructions and Quirkiness of R

```r
# 7
# a
1:10 > 5
# The result of this line is a vector with ten boolean values.
# This execution is to compare every number from 1 to 10 with 5,
# so if the number is larger than 5, the result will be True,
# and if the result is smaller then 5, the result will be False,
# since 1, 2, 3, 4, 5 are not larger than 5, the first five boolean values are False,
# and since 6, 7, 8, 9, 10 are larger than 5, the rest of the vector are True.
# b
1:(10 > 5)
# The result of this line is 1 because this execution is to show all integers from 1 to (10>5).
# And since the value of (10>5) is True which is not a numeric variable, the only integer from
# 1 to (10>5) is 1, so the result is 1.
# Same as if we execute 1:1.
```

```
> 1:10 > 5
 [1] FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
> 1:(10 > 5)
[1] 1
```

8. Case Study - Boston Housing Market

a.

```r
# 8
install.packages("GGally")
library(tree)
library(ggplot2)
library(GGally)
library(MASS)
head(Boston)
?Boston
# a
# The Boston data frame has 506 rows and 14 columns.
# Each row in the data frame represents observations of a Boston suburb or town,
# Each column represents a predictor variable of those 506 areas such as per capita crime rate,
# pupil-teacher ratio, and so on.
```

```
> ?Boston
> head(Boston)
     crim zn indus chas   nox    rm  age    dis rad tax ptratio  black lstat medv  newCrime
1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3 396.90  4.98 24.0 -2.199283
2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8 396.90  9.14 21.6 -1.563678
3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8 392.83  4.03 34.7 -1.563996
4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7 394.63  2.94 33.4 -1.489857
5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7 396.90  5.33 36.2 -1.160836
6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7 394.12  5.21 28.7 -1.525056
  newChas
1       0
2       0
3       0
4       0
5       0
6       0
```

Boston {MASS} R Documentation

Housing Values in Suburbs of Boston

Description

The `Boston` data frame has 506 rows and 14 columns.

Usage

`Boston`

Format

This data frame contains the following columns:

`crim`

per capita crime rate by town.

`zn`

proportion of residential land zoned for lots over 25,000 sq.ft.

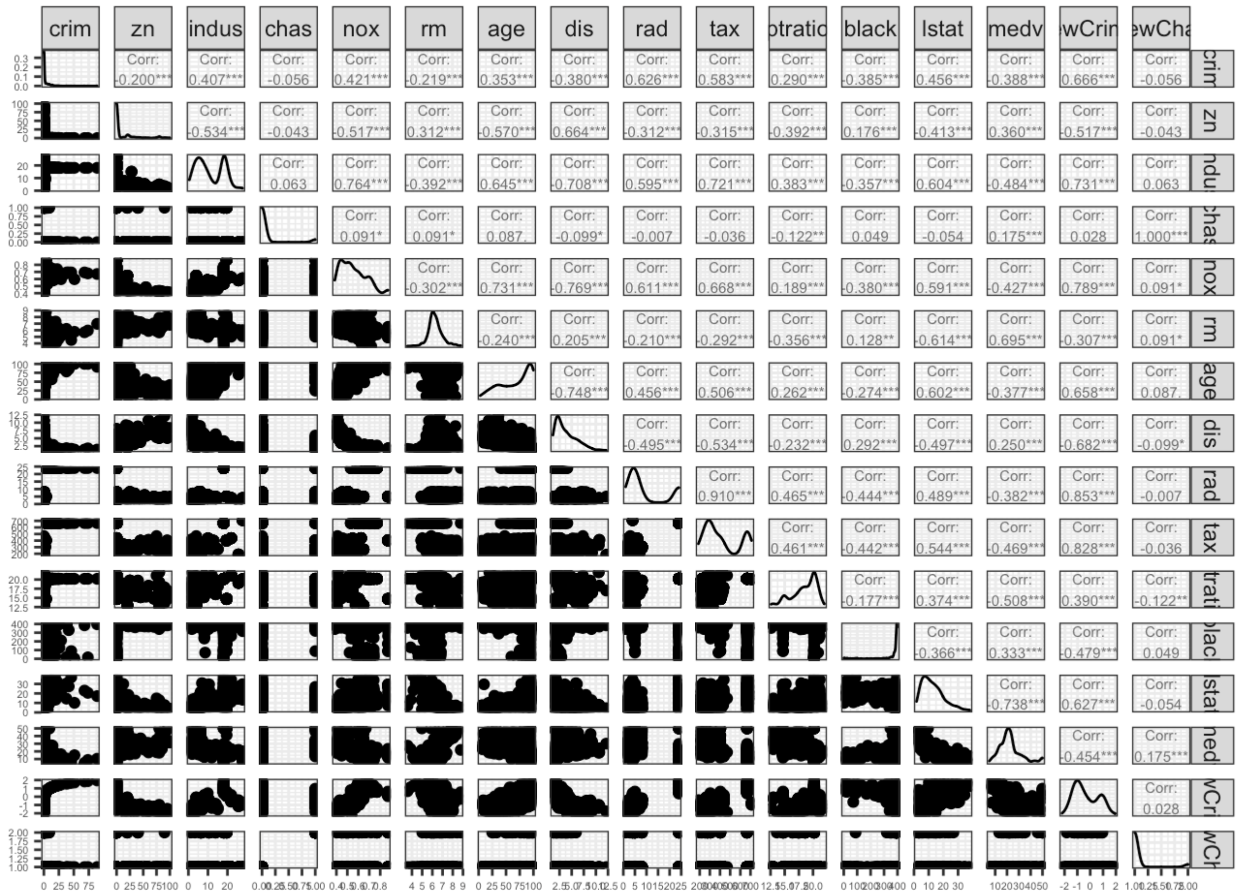
`indus`

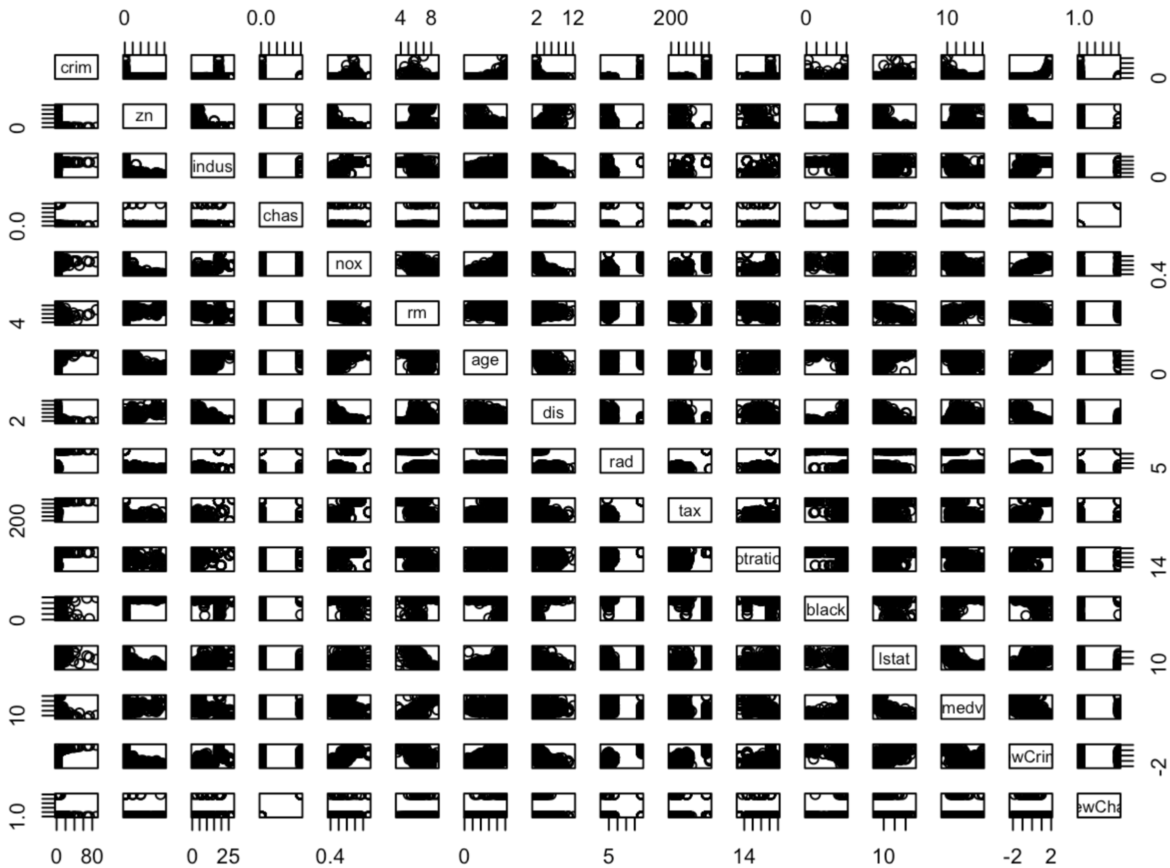
proportion of non-retail business acres per town.

b.

```
# b
ggpairs(data = Boston, title = "Boston Data Pairwise Scatterplots",
        upper = list(continuous = wrap("cor", size = 2))) +
        theme_bw() +
        theme(axis.text = element_text(size = 4))
# We can use pairs() as well
# I like to use ggpairs() rather than pairs() because ggpairs() is easier to read
# and it can show all correlation numbers of each two columns
# which is helpful to define the relationship within each column.
pairs(Boston)
# Check data type of each column
str(Boston)
# Change other data type to numeric
Boston$chas <- as.numeric(Boston$chas)
Boston$rad <- as.numeric(Boston$rad)
Boston$newChas <- as.numeric(Boston$newChas)
# Correlation matrix of Boston data frame
cor(Boston)
# From the results we can see all relationships between each two columns and we can use these
# to know which column is the most relevant to another one. For example, "rad" has the highest
# positive correlation of "crim". Also, every graph in pairs plots shows how much a column is
# relevant to another. For example, "nox" and "chas" are not very correlated since the plot is
# parallel. We also can know variable distribution on the diagonal of ggpairs plots.
# ("newCrime" and "newChas" were added for following questions.)
```

Boston Data Pairwise Scatterplots




crim
zn
indus
chas
nox
rm
age
dis
rad
tax
ptratio
black
lstat
medv
wCrim
ewCha

```
> cor(Boston)
                crim          zn       indus         chas         nox          rm         age
crim       1.00000000 -0.20046922  0.40658341 -0.055891582  0.42097171 -0.21924670  0.35273425
zn        -0.20046922  1.00000000 -0.53382819 -0.042696719 -0.51660371  0.31199059 -0.56953734
indus      0.40658341 -0.53382819  1.00000000  0.062938027  0.76365145 -0.39167585  0.64477851
chas      -0.05589158 -0.04269672  0.06293803  1.000000000  0.09120281  0.09125123  0.08651777
nox        0.42097171 -0.51660371  0.76365145  0.091202807  1.00000000 -0.30218819  0.73147010
rm        -0.21924670  0.31199059 -0.39167585  0.091251225 -0.30218819  1.00000000 -0.24026493
age        0.35273425 -0.56953734  0.64477851  0.086517774  0.73147010 -0.24026493  1.00000000
dis       -0.37967009  0.66440822 -0.70802699 -0.099175780 -0.76923011  0.20524621 -0.74788054
rad        0.62550515 -0.31194783  0.59512927 -0.007368241  0.61144056 -0.20984667  0.45602245
tax        0.58276431 -0.31456332  0.72076018 -0.035586518  0.66802320 -0.29204783  0.50645559
ptratio    0.28994558 -0.39167855  0.38324756 -0.121515174  0.18893268 -0.35550149  0.26151501
black     -0.38506394  0.17552032 -0.35697654  0.048788485 -0.38005064  0.12806864 -0.27353398
lstat      0.45562148 -0.41299457  0.60379972 -0.053929298  0.59087892 -0.61380827  0.60233853
medv      -0.38830461  0.36044534 -0.48372516  0.175260177 -0.42732077  0.69535995 -0.37695457
newCrime   0.66648575 -0.51709145  0.73082136  0.028496480  0.78861573 -0.30694282  0.65828357
newChas   -0.05589158 -0.04269672  0.06293803  1.000000000  0.09120281  0.09125123  0.08651777
                 dis          rad         tax    ptratio       black      lstat       medv
crim      -0.37967009  0.625505145  0.58276431  0.2899456 -0.38506394  0.4556215 -0.3883046
zn         0.66440822 -0.311947826 -0.31456332 -0.3916785  0.17552032 -0.4129946  0.3604453
indus     -0.70802699  0.595129275  0.72076018  0.3832476 -0.35697654  0.6037997 -0.4837252
chas      -0.09917578 -0.007368241 -0.03558652 -0.1215152  0.04878848 -0.0539293  0.1752602
nox       -0.76923011  0.611440563  0.66802320  0.1889327 -0.38005064  0.5908789 -0.4273208
rm         0.20524621 -0.209846668 -0.29204783 -0.3555015  0.12806864 -0.6138083  0.6953599
age       -0.74788054  0.456022452  0.50645559  0.2615150 -0.27353398  0.6023385 -0.3769546
dis        1.00000000 -0.494587930 -0.53443158 -0.2324705  0.29151167 -0.4969958  0.2499287
rad       -0.49458793  1.000000000  0.91022819  0.4647412 -0.44441282  0.4886763 -0.3816262
tax       -0.53443158  0.910228189  1.00000000  0.4608530 -0.44180801  0.5439934 -0.4685359
ptratio   -0.23247054  0.464741179  0.46085304  1.0000000 -0.17738330  0.3740443 -0.5077867
black      0.29151167 -0.444412816 -0.44180801 -0.1773833  1.00000000 -0.3660869  0.3334608
lstat     -0.49699583  0.488676335  0.54399341  0.3740443 -0.36608690  1.0000000 -0.7376627
medv       0.24992873 -0.381626231 -0.46853593 -0.5077867  0.33346082 -0.7376627  1.0000000
newCrime  -0.68190317  0.853406927  0.82823360  0.3895537 -0.47875518  0.6266150 -0.4543020
newChas   -0.09917578 -0.007368241 -0.03558652 -0.1215152  0.04878848 -0.0539293  0.1752602
            newCrime      newChas
crim       0.66648575 -0.055891582
zn        -0.51709145 -0.042696719
indus      0.73082136  0.062938027
chas       0.02849648  1.000000000
nox        0.78861573  0.091202807
rm        -0.30694282  0.091251225
age        0.65828357  0.086517774
dis       -0.68190317 -0.099175780
rad        0.85340693 -0.007368241
tax        0.82823360 -0.035586518
```

c.

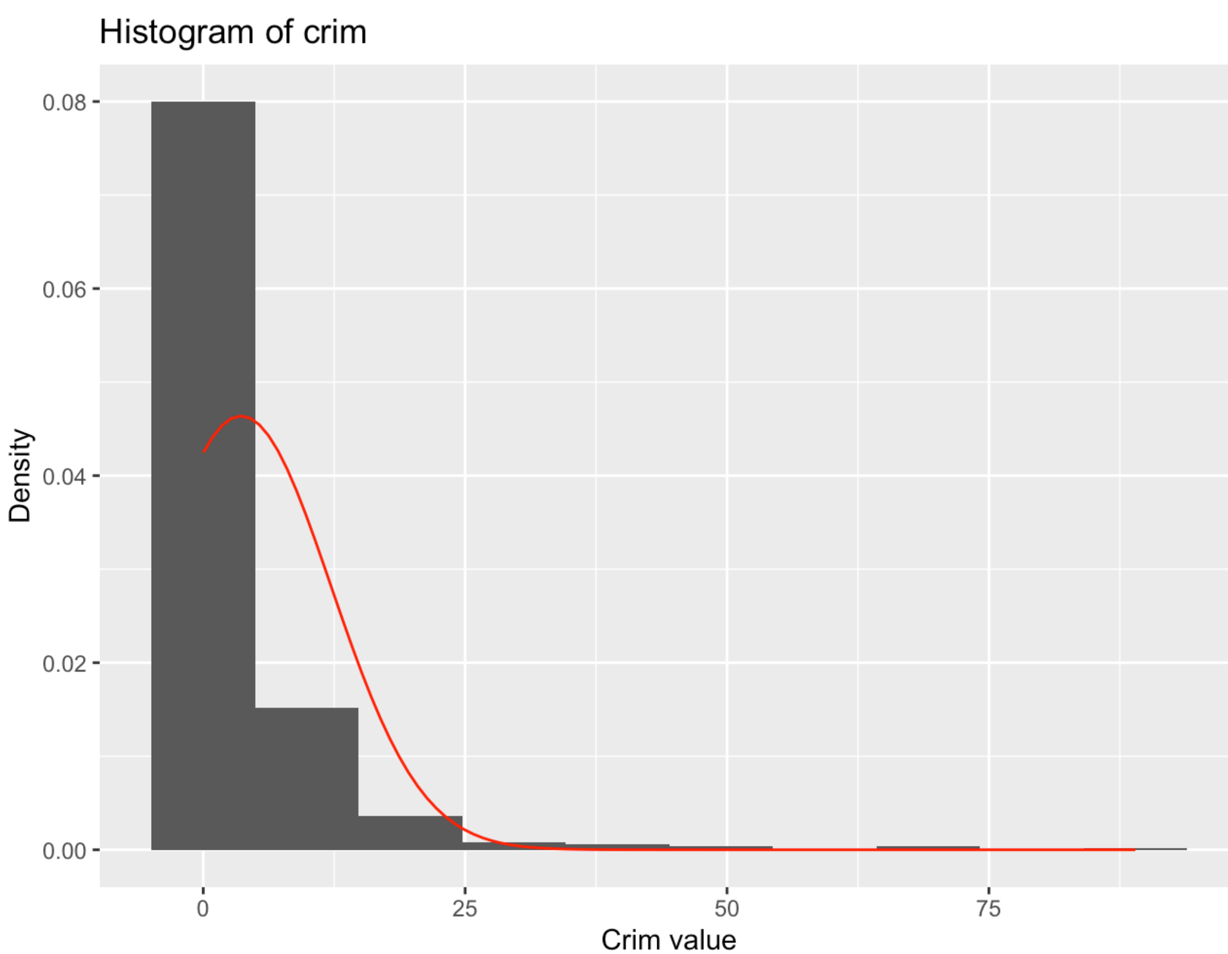
```r
# c
# Use linear regression to test if other columns have linear relationship with "crim"
fit <- lm(crim ~ ., Boston)
fit
summary(fit)
# From the linear regression summary, we can see "zn", "nox", "dis" and "medv" are
# significant for predicting "crim". And since the coefficients are provided,
# we can know how much will variable crim change when other variables change.
# From the results we got for part b, we can see all correlation values of "crim" with other
# predictors. "zn", "chas", "rm", "dis", "black", and "medv" have negative correclation
# with "crim, whereas "indus", "nox", "age", "rad", "tax", "ptratio", and "lstat" have
# positive correclation with "crim".
```

```
> fit <- lm(crim ~ ., Boston)
> fit

Call:
lm(formula = crim ~ ., data = Boston)

Coefficients:
(Intercept)           zn        indus         chas          nox           rm          age  
  28.815843     0.081738    -0.126213    -0.597332   -22.455793     0.584749    -0.017450  
        dis          rad          tax      ptratio        black        lstat         medv  
  -0.970012     0.137292    -0.003363    -0.141233    -0.002798     0.026137    -0.231933  
   newCrime     newChas1  
   7.267987           NA  

> summary(fit)

Call:
lm(formula = crim ~ ., data = Boston)

Residuals:
   Min     1Q Median     3Q    Max 
-8.122 -2.570 -0.650  1.481 67.119 

Coefficients: (1 not defined because of singularities)
             Estimate Std. Error t value Pr(>|t|)    
(Intercept)  28.815843   6.826482   4.221 2.90e-05 ***
zn            0.081738   0.017823   4.586 5.74e-06 ***
indus        -0.126213   0.077567  -1.627 0.104348    
chas         -0.597332   1.093331  -0.546 0.585079    
nox         -22.455793   5.066704  -4.432 1.15e-05 ***
rm            0.584749   0.567938   1.030 0.303704    
age          -0.017450   0.016735  -1.043 0.297574    
dis          -0.970012   0.261062  -3.716 0.000226 ***
rad           0.137292   0.095503   1.438 0.151194    
tax          -0.003363   0.004776  -0.704 0.481675    
ptratio      -0.141233   0.173306  -0.815 0.415504    
black        -0.002798   0.003443  -0.813 0.416728    
lstat         0.026137   0.071007   0.368 0.712967    
medv         -0.231933   0.056176  -4.129 4.29e-05 ***
newCrime      7.267987   0.800809   9.076  < 2e-16 ***
newChas1            NA         NA      NA       NA    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 5.965 on 491 degrees of freedom
Multiple R-squared:  0.5324,	Adjusted R-squared:  0.5191 
F-statistic: 39.94 on 14 and 491 DF,  p-value: < 2.2e-16
```

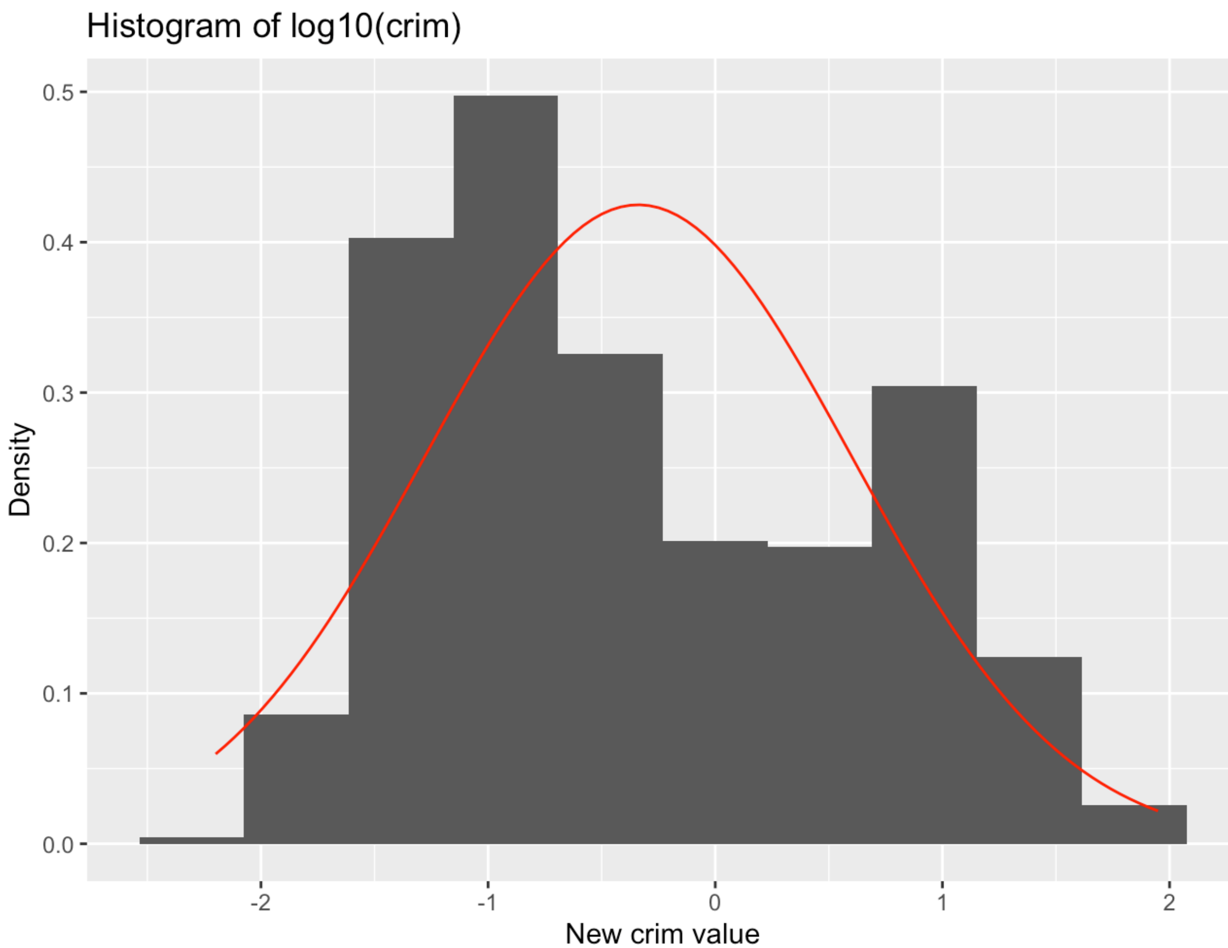
d.

```r
# d
# Calculate mean and sd of "crim"
mean(Boston$crim)
sd(Boston$crim, na.rm = FALSE)
# Histogram of "crim" with normal density function
p <- ggplot(Boston, aes(crim)) +
  geom_histogram(aes(y = ..density..), bins = 10) +
  geom_function(fun = dnorm, args = list(mean = 3.613524, sd = 8.601545), color = "red")
p + ggtitle("Histogram of crim") +
  xlab("Crim value") + ylab("Density")
# The histogram shows varible crim is not normally distributed.
# We can change "crim" values to log10(crim values) to make it more like a normal distribution.
Boston$newCrime <- log10(Boston$crim)
mean(Boston$newCrim)
sd(Boston$newCrim, na.rm = FALSE)
q <- ggplot(Boston, aes(newCrime)) +
  geom_histogram(aes(y = ..density..), bins = 10) +
  geom_function(fun = dnorm, args = list(mean = -0.3389392, sd = 0.9389665), color = "red")
q + ggtitle("Histogram of log10(crim)") +
  xlab("New crim value") + ylab("Density")
```

```
> mean(Boston$crim)
[1] 3.613524
> sd(Boston$crim, na.rm = FALSE)
[1] 8.601545
```

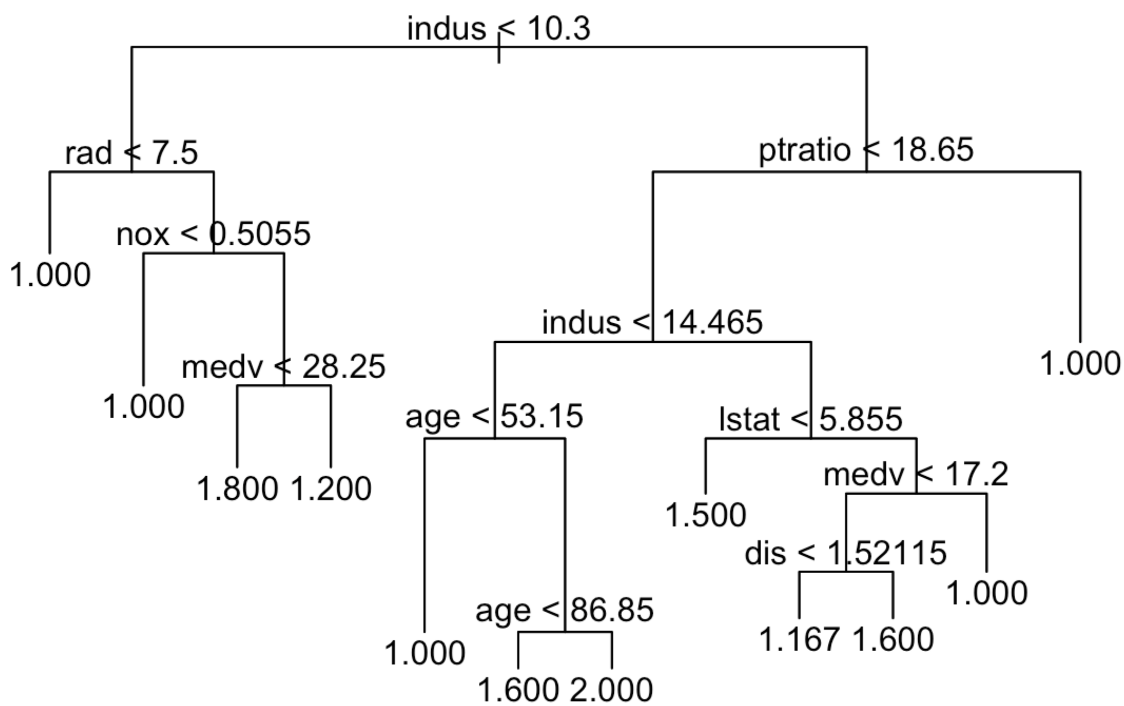


```
> Boston$newCrime <- log10(Boston$crim)
> mean(Boston$newCrim)
[1] -0.3389392
> sd(Boston$newCrim, na.rm = FALSE)
[1] 0.9389665
```



e.

```r
# e
# Change variable "chas" to a factor
# "chas" has two values: 1 represents tract bounds river, 0 otherwise.
Boston$newChas <- as.factor(Boston$chas)
# Set half of the whole data set to training data, and another half to testing data
train <- Boston[ 1:253,]
test  <- Boston[ 254:506,]
# Create a decision tree model by training data
# Set all variables except newChas in the training data to predictors
fit <- tree(newChas ~ crim + zn + indus + nox + rm + age +
              dis + rad + tax+ ptratio + black + lstat + medv, train)
plot(fit)
text(fit)
# Use testing data to test the accuracy of the prediction model
pred <- predict(fit, test, type = "class")
pred
# Create a table to evaluate model fit
tt <- table(pred, test$newChas)
tt
# Calculate the model accuracy
print((tt[1,1] + tt[2,2])/nrow(test))
```




```
> pred
  [1] 0 0 0 0 0 0 0 1 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [48] 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [95] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[142] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[189] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[236] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Levels: 0 1
```

```
> tt <- table(pred, test$newChas)
> tt

pred   0   1
   0 228  15
   1  10   0
> print((tt[1,1] + tt[2,2])/nrow(test))
[1] 0.9011858
```