Analytics 512 Homework 3

Arif Ali

02/21/16

Exercise 3.7 #10

In [1]:
```
library(ISLR)
data(Carseats)
```

Part A

In [2]:
```
Carseats.mm = lm(Sales ~ Price + Urban + US, data = Carseats)
```

Part B

In [3]:
```
summary(Carseats.mm)
```

Out[3]:
```
Call:
lm(formula = Sales ~ Price + Urban + US, data = Carseats)

Residuals:
    Min      1Q  Median      3Q     Max 
-6.9206 -1.6220 -0.0564  1.5786  7.0581 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) 13.043469   0.651012  20.036  < 2e-16 ***
Price       -0.054459   0.005242 -10.389  < 2e-16 ***
UrbanYes    -0.021916   0.271650  -0.081    0.936    
USYes        1.200573   0.259042   4.635 4.86e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.472 on 396 degrees of freedom
Multiple R-squared:  0.2393,    Adjusted R-squared:  0.2335 
F-statistic: 41.52 on 3 and 396 DF,  p-value: < 2.2e-16
```

Price: Price has a significiant effect on the Response as evidence by its p-value. However, the coeefficient estimate is negative, which indicates a negative relationship.

UrbanYes: No relation is indicated based on the very high p-value. Based on a summary, UrbanYes is a qualitative variable indicating if a store is urban based or not.

USYes: USYes has a significiant effect on the Response as evidence by its p-value. The coefficient estimate indicates a positive relationship. USYes is a qualitative variable indicating if a store is based on the US.

Part C

$Sales = 13.043469 - 0.054459 * Price - 0.021916 * UrbanYes + 1.200573 * USYes$

Part D

Price and USYes based on the p-value.

Part E

In [4]:
```
Carseats.mm.bs = lm(Sales ~ Price + US, data = Carseats)
summary(Carseats.mm.bs)
```

Out[4]:
```
Call:
lm(formula = Sales ~ Price + US, data = Carseats)

Residuals:
    Min      1Q  Median      3Q     Max 
-6.9269 -1.6286 -0.0574  1.5766  7.0515 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 13.03079    0.63098  20.652  < 2e-16 ***
Price       -0.05448    0.00523 -10.416  < 2e-16 ***
USYes        1.19964    0.25846   4.641 4.71e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.469 on 397 degrees of freedom
Multiple R-squared:  0.2393,	Adjusted R-squared:  0.2354 
F-statistic: 62.43 on 2 and 397 DF,  p-value: < 2.2e-16
```

Part G

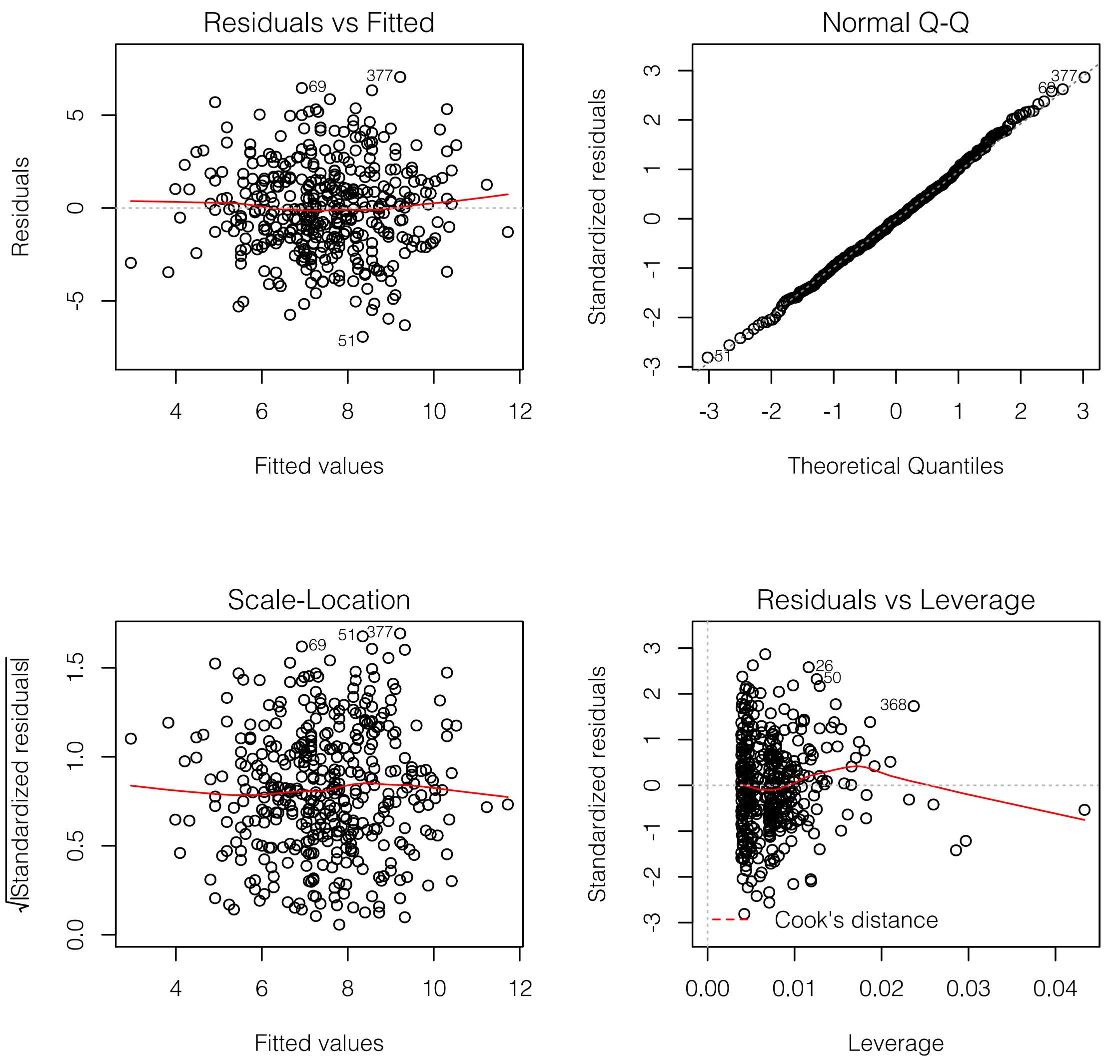
In [5]:
```
confint(Carseats.mm.bs)
```

Out[5]:

	2.5 %	97.5 %
(Intercept)	11.79032	14.27127
Price	-0.06475984	-0.04419543
USYes	0.6915196	1.7077663

Part H

In [6]:
```
par(mfrow = c(2,2))
plot(Carseats.mm.bs)
```



In terms of leverage, one of the points does seem to have a pull indicating high leverage. However, there doesn't seem to be any noticable outliers from the Residuals vs Fitted or the $\sqrt{Standardized\ Residuals}$ vs Fitted Values. While there are some values that have slightly higher $\sqrt{Standardized\ Residuals}$ or Residual values.

Exercise 4.7 #2

$$p_k(x) = \frac{\pi_k * \frac{1}{\sqrt{2\pi\sigma}} exp\left\{-\frac{1}{2\sigma^2}(x-u_k)^2\right\}}{\sum_{i=1}^{K} \pi_i * \frac{1}{\sqrt{2\pi\sigma}} exp\left\{-\frac{1}{2\sigma^2}(x-u_i)^2\right\}}$$
$$\delta_k(x) = x * \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + log(\pi_k)$$

$\frac{1}{\sqrt{2\pi\sigma}} exp\left\{-\frac{1}{2\sigma^2}(x-u_i)^2\right\}$ is constant for all k values, therefore, it does not factor into maximizing the entire function.

$log\left(\pi_k * \frac{1}{\sqrt{2\pi\sigma}} exp\left\{-\frac{1}{2\sigma^2}(x-u_k)^2\right\}\right) = log(\pi_k) + log\left(\frac{1}{\sqrt{2\pi\sigma}} exp\left\{-\frac{1}{2\sigma^2}(x-u_k)^2\right\}\right) = log(\pi_k) + log(\frac{1}{\sqrt{2\pi\sigma}}) * (-\frac{1}{2\sigma^2}(x-u_k)^2)$

$log(\frac{1}{\sqrt{2\pi\sigma}})$ is a constant for all numerators that does not change between different k values; therefore, also should not factor into the maximization of the function. $log(\pi_k) + -\frac{1}{2\sigma^2}\left(x^2 - 2xu_k + \mu_k^2\right)$

x^2 doesn't change based on different k-values, and therefore shouldn't factor into the maximizing function.

$log(\pi_k) + -\frac{1}{2\sigma^2}\left(-2xu_k + \mu_k^2\right) = log(\pi_k) - \frac{1}{2\sigma^2} * -2xu_k - \frac{1}{2\sigma^2} * \mu_k^2$

This is the same as:

$$\delta_k(x) = x * \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + log(\pi_k)$$

Exercise 4.7 #5

Part A

For the training sets, QDA would perform better because the method is more flexible and could overcome issues of complexity within the training. For the testing sets, we would expect LDA to perform better than QDA if the Bayes Decision boundary is Linear because QDA might overfit on the training data set.

Part B

For the training and testing sets, we would expect QDA to perform better than LDA if the Bayes Decision boundary is non-Linear. This is because the non-Linear Bayes Decision boundary indicates great complexity within the data requiring the use of a more flexible method.

Part C

Given an extremely small n, it's possible that the QDA would be a better alternative to LDA because the Linearity of the Bayes Decision boundary could be questionable. However, this cannot be concluded as a general case.

Part D

False, QDA could overfit on the training data set which would affect the error rate on the test data set.

Exercise 4.7 #6

Part A

$$Y = -6 + 0.05 * Hours + GPA$$
$$p(X) = \frac{e^{-6+0.05*Hours+GPA}}{1 + e^{-6+0.05*Hours+GPA}}$$

In [7]:
```
exp(-6 + 0.05*40 + 3.5)/(1+exp(-6 + 0.05*40 + 3.5))
```

Out[7]: 0.377540668798145

Part B

$$\frac{e^{-6+0.05*Hours+3.5}}{1 + e^{-6+0.05*Hours+3.5}} > 0.5 \implies$$
$$e^{-6+0.05*Hours+3.5} > 0.5 * (1 + e^{-6+0.05*Hours+3.5}) \implies$$
$$e^{-6+0.05*Hours+3.5} * 0.5 > 0.5 \implies$$
$$e^{-6+0.05*Hours+3.5} > 1 \implies$$
$$-6 + 0.05 * Hours + 3.5 > log(1) \implies$$
$$0.05 * Hours > 0 + 2.5 \implies Hours > 2.5/0.05 = 50$$

Exercise 4.7 #11

Part A

In [8]:
```
library(ISLR)
data(Auto)
Auto$mpg01 = rep(0, times = nrow(Auto))
Auto$mpg01[Auto$mpg>median(Auto$mpg)] = 1
```

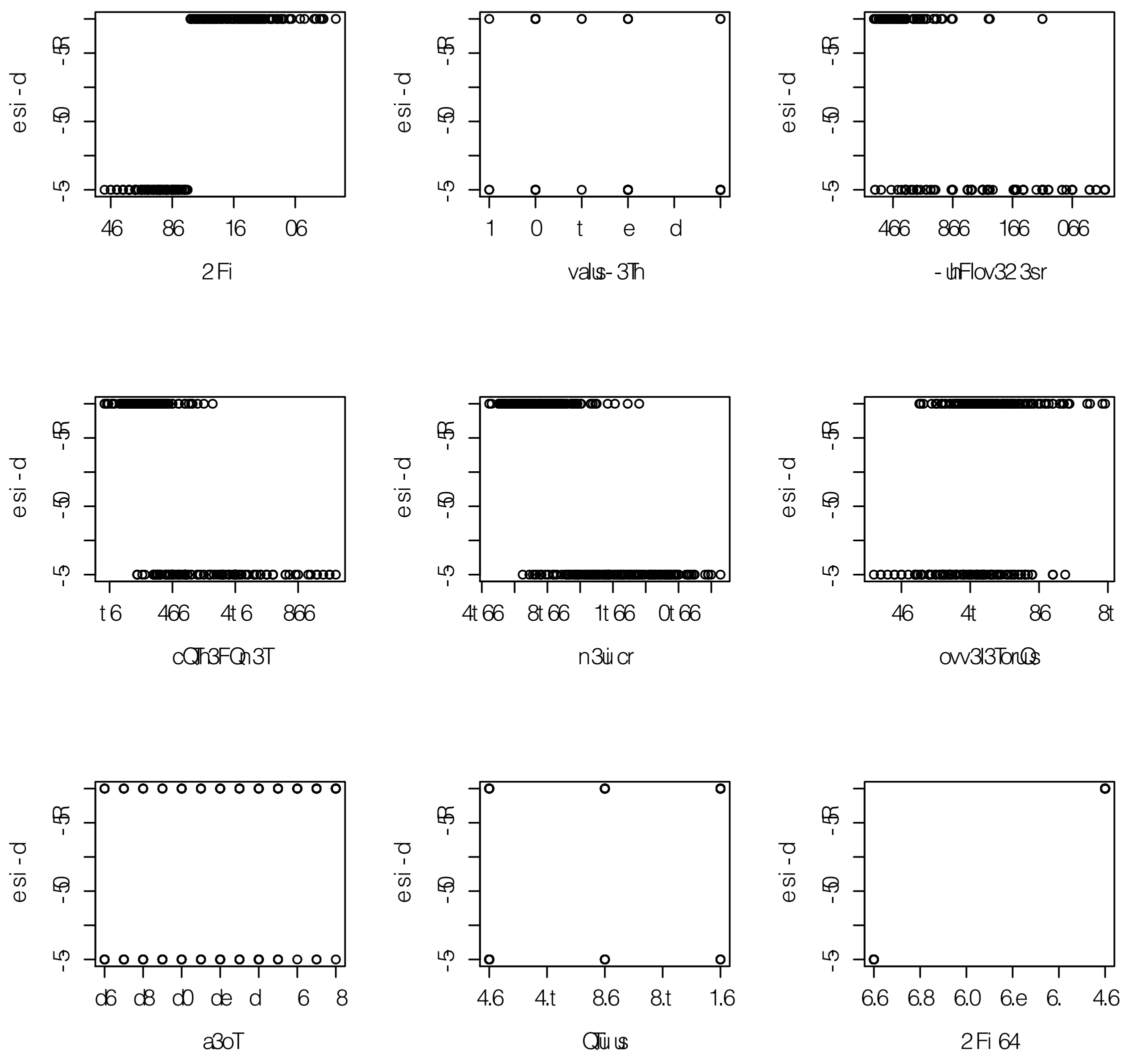
In [9]:
```
cor(Auto[,-9])[,ncol(Auto[,-9])]
```

Out[9]:

mpg	0.836939229153618
cylinders	-0.75919388654222
displacement	-0.753476593523588
horsepower	-0.667052581047159
weight	-0.75775657142698
acceleration	0.346821530080934
year	0.429904226574927
origin	0.513698448318672
mpg01	1

Part B

```
In [10]: par(mfrow = c(3,3))
         for(i in names(Auto[,-9])){
             plot(Auto[,i], Auto[,"mpg01"],xlab = i, ylab = "mpg01")
         }
```



I couldn't really tell that much from the scatterplots, so I looked the correlations. From that mpg01 has best relation between cylinders, displacement, horsepower, and weight. It seemed that mpg while highly correlated should not be included in attempting to predict mpg01. Looking at the graphs for these corresponding variables, there seems to be some patterns associated between the two binary values for mpg01. The overlaps could indicate why the correlation values are not exactly -1 or 1.

Part C

```
In [11]: training = sample(x = nrow(Auto), size = 0.5*nrow(Auto),replace = F)
         training.set = Auto[training,]
         test.set = Auto[-training,-10]
```

Part D

In [12]:
```
library(MASS)
lda.auto = lda(mpg01~cylinders+displacement+horsepower+weight, data = training.set[,-
9])
```

In [13]:
```
lda.predict = predict(lda.auto, test.set)
mean(lda.predict$class != Auto[-training,10])
```

Out[13]: 0.0969387755102041

Part E

In [14]:
```
library(MASS)
qda.auto = qda(mpg01~cylinders+displacement+horsepower+weight, data = training.set[,-
9])
```

In [15]:
```
qda.predict = predict(qda.auto, test.set)
mean(qda.predict$class != Auto[-training,10])
```

Out[15]: 0.11734693877551

Part F

In [16]:
```
logistic.regrssion.auto = glm(mpg01~cylinders+displacement+horsepower+weight,
                              data = training.set[,-9], family = binomial)
```

In [17]:
```
logistic.regrssion.predict = predict(logistic.regrssion.auto, test.set,type="response")
logistic.regrssion.pred = rep(0, times = length(logistic.regrssion.predict))
logistic.regrssion.pred[logistic.regrssion.predict>0.5] = 1
mean(logistic.regrssion.pred != Auto[-training,10])
```

Out[17]: 0.112244897959184

Part G

In [18]:
```
ks = 1:20
library(FNN)
k.results = data.frame(ks, error = ks)
for(i in ks){
    knn.predict = knn(train = training.set[,c("cylinders","displacement","horsepowe
r","weight")],
        test = test.set[,c("cylinders","displacement","horsepower","weight")], cl = tra
ining.set$mpg01, k = i)
    k.results[k.results$ks == i, "error"]=mean(knn.predict != Auto[-training,10])
}
k.results
```

Out[18]:

	ks	error
1	1	0.1428571
2	2	0.127551
3	3	0.122449
4	4	0.1326531
5	5	0.1173469
6	6	0.122449
7	7	0.127551
8	8	0.122449
9	9	0.127551
10	10	0.1173469
11	11	0.127551
12	12	0.1173469
13	13	0.127551
14	14	0.1173469
15	15	0.1326531
16	16	0.127551
17	17	0.127551
18	18	0.127551
19	19	0.127551
20	20	0.127551

The k values with the lowest error is k = 5.