Assignment 3 Chapter 4

(4.a)

10%, except x<0.05 x>0.95

(4.b)

It is an area concept, 10% multiplied by 10%, which means we only got 1% availability.

(4.c)

10% to the power of 100, roughly 0.

(4.d)

It seems that as p (dimensionality) increases, the percentage of available observations in the specific range decrease expontentially.

(4.e)

P=1 l=0.1, p=2 l=square root of 0.01, p=100 l=100th root of 0.01

(5.a)

QDA perform better on the training set since it will overfit the linear boundary, whose MSE will be nearly 0. LDA perform better both on the test set since the variance and bias of it are both so low.

(5.b)

QDA can perform better on both on the set, training and test. Since the LDA will have a very high bias for the two data set.

(5.c)

QDA will perform better, for this large n sample size, QDA is easier to capture the complicated relationship between the n compared to LDA, which mean QDA can provide the better fit.

(5.d)

False. QDA will always be too flexible to overfit the noise in the data. even if the variance of the data is small, which will lead QDA (flexible method) to perform well, the LDA will still be better, since the true boudary is a line….

(6.a)

$x1=40$ $x2=3.5$ $\beta0=-6,\beta1=0.05,\beta2=1$ $p(X)=exp(\beta0+\beta1X1+\beta2X2)/(1+exp(\beta0+\beta1X1+\beta2X2)=37.75\%$

(6.b)

$p(X)=0.5$, $x2=3.5$ $\beta0=-6,\beta1=0.05,\beta2=1$ plug in the equation above, and solve it, $x1=50$hrs

(8)

logistic regression 20% training error 30% testing error 1-nearest neighbors average 18%, 0% for training error and 36% for testing error 1-nearest neighbors is too flexible to overfit all the noise, 0% for training error. I will go for logistic regression since it got a lower testing error.

(10.a)

```r
library(ISLR)
data("Weekly")
summary(Weekly)
```

```
##       Year          Lag1               Lag2               Lag3         
##  Min.   :1990   Min.   :-18.1950   Min.   :-18.1950   Min.   :-18.1950  
##  1st Qu.:1995   1st Qu.: -1.1540   1st Qu.: -1.1540   1st Qu.: -1.1580  
##  Median :2000   Median :  0.2410   Median :  0.2410   Median :  0.2410  
##  Mean   :2000   Mean   :  0.1506   Mean   :  0.1511   Mean   :  0.1472  
##  3rd Qu.:2005   3rd Qu.:  1.4050   3rd Qu.:  1.4090   3rd Qu.:  1.4090  
##  Max.   :2010   Max.   : 12.0260   Max.   : 12.0260   Max.   : 12.0260  
##       Lag4               Lag5              Volume       
##  Min.   :-18.1950   Min.   :-18.1950   Min.   :0.08747  
##  1st Qu.: -1.1580   1st Qu.: -1.1660   1st Qu.:0.33202  
##  Median :  0.2380   Median :  0.2340   Median :1.00268  
##  Mean   :  0.1458   Mean   :  0.1399   Mean   :1.57462  
##  3rd Qu.:  1.4090   3rd Qu.:  1.4050   3rd Qu.:2.05373  
##  Max.   : 12.0260   Max.   : 12.0260   Max.   :9.32821  
##      Today          Direction 
##  Min.   :-18.1950   Down:484  
##  1st Qu.: -1.1540   Up  :605  
##  Median :  0.2410             
##  Mean   :  0.1499             
##  3rd Qu.:  1.4050             
##  Max.   : 12.0260             
```

```r
cor(Weekly[, -9])
```

```
##               Year         Lag1        Lag2        Lag3         Lag4
## Year    1.00000000 -0.032289274 -0.03339001 -0.03000649 -0.031127923
## Lag1   -0.03228927  1.000000000 -0.07485305  0.05863568 -0.071273876
## Lag2   -0.03339001 -0.074853051  1.00000000 -0.07572091  0.058381535
## Lag3   -0.03000649  0.058635682 -0.07572091  1.00000000 -0.075395865
## Lag4   -0.03112792 -0.071273876  0.05838153 -0.07539587  1.000000000
## Lag5   -0.03051910 -0.008183096 -0.07249948  0.06065717 -0.075675027
## Volume  0.84194162 -0.064951313 -0.08551314 -0.06928771 -0.061074617
## Today  -0.03245989 -0.075031842  0.05916672 -0.07124364 -0.007825873
##                Lag5      Volume        Today
## Year   -0.030519101  0.84194162 -0.032459894
## Lag1   -0.008183096 -0.06495131 -0.075031842
## Lag2   -0.072499482 -0.08551314  0.059166717
## Lag3    0.060657175 -0.06928771 -0.071243639
## Lag4   -0.075675027 -0.06107462 -0.007825873
## Lag5    1.000000000 -0.05851741  0.011012698
## Volume -0.058517414  1.00000000 -0.033077783
## Today   0.011012698 -0.03307778  1.000000000
```

Volume and year have a positive relationship.

(10.b)

```r
attach(Weekly)
glm.ex10=glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume, data = Weekly, family = binomial)
summary(glm.ex10)
```

```
## 
## Call:
## glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + 
##     Volume, family = binomial, data = Weekly)
## 
## Deviance Residuals: 
##     Min       1Q   Median       3Q      Max  
## -1.6949  -1.2565   0.9913   1.0849   1.4579  
## 
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)   
## (Intercept)  0.26686    0.08593   3.106   0.0019 **
## Lag1        -0.04127    0.02641  -1.563   0.1181   
## Lag2         0.05844    0.02686   2.175   0.0296 * 
## Lag3        -0.01606    0.02666  -0.602   0.5469   
## Lag4        -0.02779    0.02646  -1.050   0.2937   
## Lag5        -0.01447    0.02638  -0.549   0.5833   
## Volume      -0.02274    0.03690  -0.616   0.5377   
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
##     Null deviance: 1496.2  on 1088  degrees of freedom
## Residual deviance: 1486.4  on 1082  degrees of freedom
## AIC: 1500.4
## 
## Number of Fisher Scoring iterations: 4
```

Lag2 a little bit significance.

(10.c)

```r
glm.pre10= predict(glm.ex10, type = "response")
glm.pre=rep("Down", 1089)
glm.pre[glm.pre10 > 0.5] = "Up"
table(glm.pre, Weekly$Direction)
```

```
## 
## glm.pre Down  Up
##    Down   54  48
##    Up    430 557
```

```r
mean(glm.pre == Weekly$Direction)
```

```
## [1] 0.5610652
```

The total correct percentage is 56.1%. The UP correct percentage is 92.1%. The Down correct percentage is 11.2%.

(10.d)

```r
train = (Year<2009)
test.data = Weekly[!train, ]
glm.d = glm(Direction ~ Lag2, data = Weekly, family = binomial, subset = train)
glm.logis = predict(glm.d, test.data, type = "response")
glm.prelog=rep("Down", length(glm.logis))
glm.prelog[glm.logis > 0.5] = "Up"
Direction0910 = Direction[!train]
table(glm.prelog, Direction0910)
```

```
##           Direction0910
## glm.prelog Down Up
##       Down    9  5
##       Up     34 56
```

```r
mean(glm.prelog == Direction0910)
```

```
## [1] 0.625
```

(10.e)

```r
library(MASS)
lda.e=lda(Direction~Lag2, data=Weekly, subset = train)
lda.predict= predict(lda.e, test.data)
table(lda.predict$class, Direction0910)
```

```
##       Direction0910
##        Down Up
##   Down    9  5
##   Up     34 56
```

```r
mean(lda.predict$class == Direction0910)
```

```
## [1] 0.625
```

(10.f)

```r
qda.f=qda(Direction ~ Lag2, data = Weekly, subset = train)
qda.predict= predict(qda.f, test.data)$class
table(qda.predict, Direction0910)
```

```
##            Direction0910
## qda.predict Down Up
##        Down    0  0
##        Up     43 61
```

```r
mean(qda.predict == Direction0910)
```

```
## [1] 0.5865385
```

(10.g)

```r
library(class)
train.X = as.matrix(Lag2[train])
test.X = as.matrix(Lag2[!train])
train.Direction = Direction[train]
set.seed(1)
knn.pred = knn(train.X, test.X, train.Direction, k = 1)
table(knn.pred, Direction0910)
```

```
##         Direction0910
## knn.pred Down Up
##     Down   21 30
##     Up     22 31
```

```r
mean(knn.pred == Direction0910)
```

```
## [1] 0.5
```

(10.h)

Logistic and LDA provide the best test error, 62.5%

(10.i)

```r
logistic = glm(Direction~ Lag1:Lag3, data=Weekly, family = binomial, subset= train)
logistic.prob=predict(logistic, test.data, type="response")
logistic.pred=rep("Down", length(logistic.prob))
logistic.pred[logistic.prob>.5]="Up"
table(logistic.pred, Direction0910)
```

```
##              Direction0910
## logistic.pred Down Up
##          Down    3  3
##          Up     40 58
```

```r
mean(logistic.pred == Direction0910)
```

```
## [1] 0.5865385
```

```r
lda.i= lda(Direction~ Lag2+I(Lag1^2), data = Weekly, subset = train)
lda.pred=predict(lda.i, test.data)$class
table(lda.pred, Direction0910)
```

```
##         Direction0910
## lda.pred Down Up
##     Down    8  2
##     Up     35 59
```

```r
mean(lda.pred == Direction0910)
```

```
## [1] 0.6442308
```

```r
qda.i= qda(Direction~ I(Lag1^2)+Lag2:Volume, data=Weekly, subset = train)
qda.pred=predict(qda.i, test.data)$class
table(qda.pred, Direction0910)
```

```
##         Direction0910
## qda.pred Down Up
##     Down   29 31
##     Up     14 30
```

```r
mean(qda.pred==Direction0910)
```

```
## [1] 0.5673077
```

```r
knn.pred1=knn(train.X, test.X, train.Direction, k=1)
table(knn.pred, Direction0910)
```

```
##         Direction0910
## knn.pred Down Up
##     Down   21 30
##     Up     22 31
```

```r
mean(knn.pred==Direction0910)
```

```
## [1] 0.5
```

```r
knn.pred100=knn(train.X, test.X, train.Direction, k=100)
table(knn.pred1, Direction0910)
```

```
##          Direction0910
## knn.pred1 Down Up
##      Down   21 30
##      Up     22 31
```

```r
mean(knn.pred1==Direction0910)
```

```
## [1] 0.5
```

As the above shows, the LDA got the best performance among the others, 64% correct rate. And from the knn, the k100 perform better than k1. All of these indicates that the linear relationship seems perfrom a little better than the more flexible one.

(11.a)

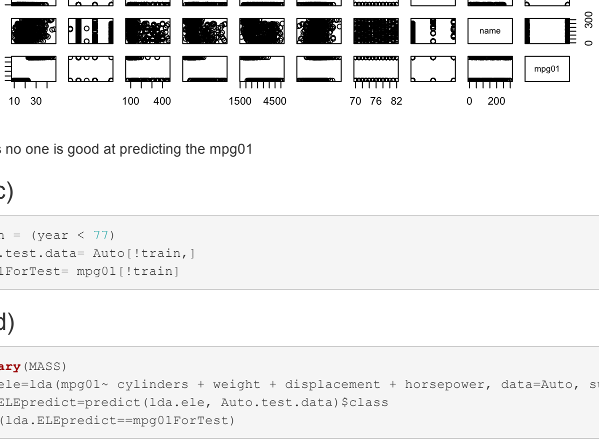
```r
library(ISLR)
data("Auto")
attach(Auto)
med=median(mpg)
mpg01=rep(0, length(Auto$mpg))
mpg01[mpg > med]= 1
Auto= data.frame(Auto, mpg01)
```

(11.b)

```r
cor(Auto[, -9])
```

```
##                     mpg  cylinders displacement horsepower     weight
## mpg           1.0000000 -0.7776175   -0.8051269 -0.7784268 -0.8322442
## cylinders    -0.7776175  1.0000000    0.9508233  0.8429834  0.8975273
## displacement -0.8051269  0.9508233    1.0000000  0.8972570  0.9329944
## horsepower   -0.7784268  0.8429834    0.8972570  1.0000000  0.8645377
## weight       -0.8322442  0.8975273    0.9329944  0.8645377  1.0000000
## acceleration  0.4233285 -0.5046834   -0.5438005 -0.6891955 -0.4168392
## year          0.5805410 -0.3456474   -0.3698552 -0.4163615 -0.3091199
## origin        0.5652088 -0.5689316   -0.6145351 -0.4551715 -0.5850054
## mpg01         0.8369392 -0.7591939   -0.7534766 -0.6670526 -0.7577566
##              acceleration       year     origin      mpg01
## mpg             0.4233285  0.5805410  0.5652088  0.8369392
## cylinders      -0.5046834 -0.3456474 -0.5689316 -0.7591939
## displacement   -0.5438005 -0.3698552 -0.6145351 -0.7534766
## horsepower     -0.6891955 -0.4163615 -0.4551715 -0.6670526
## weight         -0.4168392 -0.3091199 -0.5850054 -0.7577566
## acceleration    1.0000000  0.2903161  0.2127458  0.3468215
## year            0.2903161  1.0000000  0.1815277  0.4299042
## origin          0.2127458  0.1815277  1.0000000  0.5136984
## mpg01           0.3468215  0.4299042  0.5136984  1.0000000
```

```r
pairs(Auto)
```



From the scatter plot, it seems that there is no one is good at predicting the mpg01

(11.c)

```r
train = (year < 77)
Auto.test.data= Auto[!train,]
mpg01ForTest= mpg01[!train]
```

(11.d)

```r
library(MASS)
lda.ele=lda(mpg01~ cylinders + weight + displacement + horsepower, data=Auto, subset=train)
lda.ELEpredict=predict(lda.ele, Auto.test.data)$class
mean(lda.ELEpredict==mpg01ForTest)
```

```
## [1] 0.8932584
```

(11.e)

```r
qda.fit = qda(mpg01 ~ cylinders + weight + displacement + horsepower, data = Auto, subset = train)
qda.ELEpredict=predict(qda.fit, Auto.test.data)$class
mean(qda.ELEpredict==mpg01ForTest)
```

```
## [1] 0.8651685
```

(11.f)

```r
logis.fit = glm(mpg01 ~ cylinders + weight + displacement + horsepower, data = Auto, family = binomial, subset
= train)
logis.ELEpredict=predict(logis.fit, Auto.test.data, type="response")
log.predict=rep(0, length(logis.ELEpredict))
log.predict[logis.ELEpredict>.5]=1
mean(log.predict==mpg01ForTest)
```

```
## [1] 0.7808989
```

(11.g)

```r
library(class)
train.y=cbind(cylinders, weight, displacement, horsepower)[train, ]
test.y=cbind(cylinders, weight, displacement, horsepower)[!train, ]
train.mpg01=mpg01[train]
set.seed(1)
knn.pred3=knn(train.y, test.y, train.mpg01, k=3)
mean(knn.pred3 == mpg01ForTest)
```

```
## [1] 0.8258427
```

```r
knn.pred10=knn(train.y, test.y, train.mpg01, k=10)
mean(knn.pred10 == mpg01ForTest)
```

```
## [1] 0.8314607
```

```r
knn.pred101=knn(train.y, test.y, train.mpg01, k=101)
mean(knn.pred101 == mpg01ForTest)
```

```
## [1] 0.8202247
```

k=10 perform best!