# HW3

*Ramtin Boustani - SUID# 05999261*

## Problem 1

**Chapter 4, Exercise 4**

**(a)**

#P=1
~10%

**(b)**

#P=2
10% * 10% ~ 0.1%

**(c)**

#P=100
$10\%^{100} \sim 0\%$

**(d)**

#p= $\infty$
$\lim_{p \to \infty} (10\%)^p \sim 0\%$

**(e)**

#p=1 –> l=0.10
#p=2 –> l= $0.10^{1/2}$
#p=3 –> l= $0.10^{1/3}$
....
#p=n –> l= $0.10^{1/n}$

## Problem 2

**Chapter 4, Exercise 5**

**(a)**

Linear Bayes decision boundary
training set: QDA perform better because is more flexible
test set: LDA perform better becuase QDA could overfit the linearity

**(b)**

non-linear Bayes decision boundary
training set: QDA perform better because it is more flexible
test set: QDA perform better because it is more flexible

### (c)

Dependes on the linearity of Bayes decision boundary.
Generally, QDA perform better because of being more flexible and large size of n helps to decrease high variance issue

### (d)

False
Since QDA is more flexible and captures all noises so overfitting happens and will have lower test error rate.

# Problem 3

**Chapter 4, Exercise 6**

### (a)

```
b0= -6
b1 = 0.05
b2= 1
x1=40
x2=3.5
f=b0+(b1*x1)+(b2*x2)
p= exp(f)/(1+exp(f))
p
```

```
## [1] 0.3775407
```

37.75%

### (b)

p=0.5
$e^f = (0.5)(1 + e^f)$
$0.5 * e^f = 0.5$
$e^f = 1$
$ln(1) = f$
$0 = -6 + b1 * 0.05 + 1 * 3.5$
$b1 = 50$
Need to study 50 Hours

# Problem 4

**Chapter 4, Exercise 8**

Logistic Regression
For KNN for K=1 since test and training samples are the same so training error is zero.
Because For k=1 we choose 1st closing training sample that is itself.
Since training error is 0% and given 18% average error rate we have the following:
(0% + test-error% )/ 2 = 18% –> KNN k=1 error rate was 36%

Logestic regression has lower error rate of 30%!

# Problem 5

### Chapter 4, Exercise 10

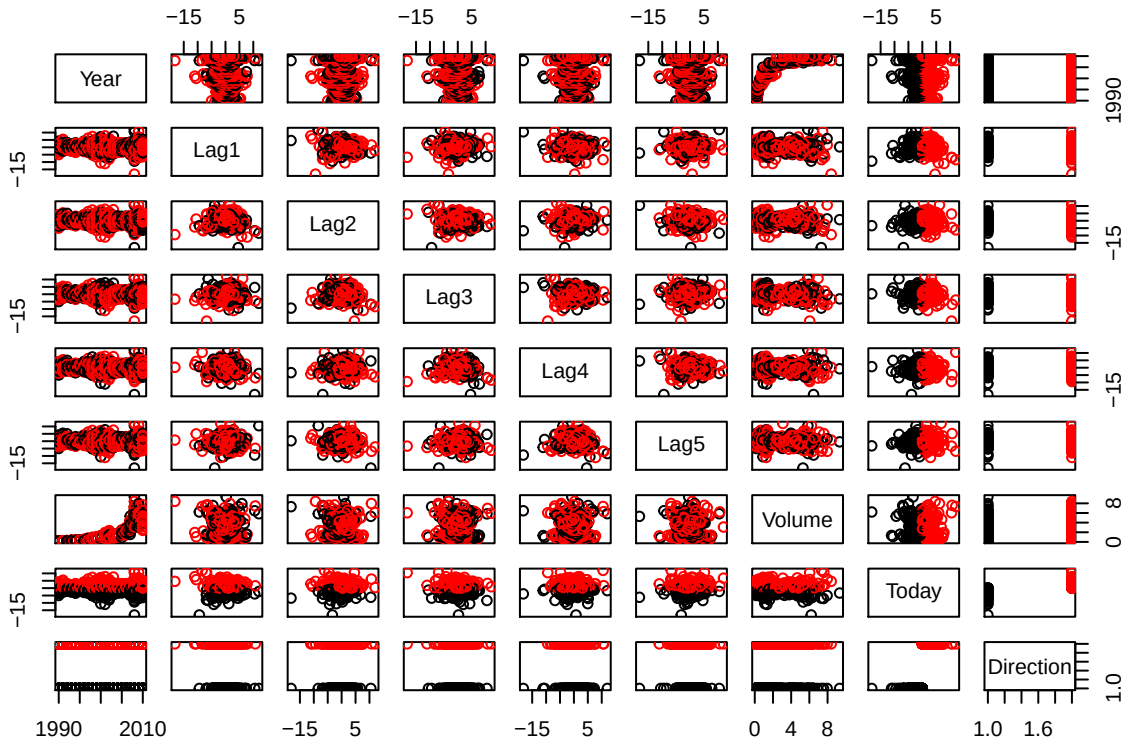
```
require(ISLR)
```

```
## Loading required package: ISLR
```

```
attach(Weekly)
```

### (a)

```
pairs(Weekly, col=Weekly$Direction )
```




```
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
```

Year and Volume have a relationship because of covariance of (0.84)

**(b)**

```
glm.fit = glm(Direction ~. - Year - Today , data = Weekly, family = binomial)
summary(glm.fit)
```

```
##
## Call:
## glm(formula = Direction ~ . - Year - Today, 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 has p-value less than 5% and statistical significance

```
glm.probs = predict(glm.fit, type = "response")
mean(glm.probs)
```

```
## [1] 0.5555556
```

Not strong prediction because of probability of being up or down is around ~50%

**(c)**

```
glm.pred = ifelse( glm.probs>0.5, "Up", "Down" )
table(glm.pred, Direction)
```

```
##         Direction
## glm.pred Down  Up
##     Down   54  48
```

```
##     Up    430 557
```

correct predication: 611/1089 = 56%
accuracy of UP predication: 557/605 = 92%
accuracy of Down predication: 54/484 = 11% (weak prediction)

### (d)

```
train = Year<=2008
glm.fit = glm(Direction~Lag2, subset=train, family = binomial)
glm.probs = predict(glm.fit, newdata = Weekly[!train,], type="response")
glm.pred = ifelse(glm.probs>0.5, "Up", "Down")
table(glm.pred, Weekly[!train,]$Direction)
```

```
##
## glm.pred Down Up
##     Down    9  5
##     Up     34 56
```

```
mean(glm.pred == Weekly[!train,]$Direction)
```

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

### (e)

Importing MASS for lda library

```
library(MASS)
```

```
lda.fit = lda(Direction ~ Lag2, data = Weekly, subset = train)
lda.pred = predict(lda.fit , newdata= Weekly[!train,], type="response")
table(lda.pred$class, Weekly[!train,]$Direction)
```

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

```
mean(lda.pred$class==Weekly[!train,]$Direction)
```

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

### (f)

```
qda.fit = qda(Direction ~ Lag2, data = Weekly, subset = train)
qda.pred = predict(qda.fit , newdata= Weekly[!train,], type="response")
table(qda.pred$class, Weekly[!train,]$Direction)
```

```
##
##        Down Up
##   Down    0  0
##   Up     43 61
```

```
mean(qda.pred$class==Weekly[!train,]$Direction)
```

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

**(g)**

```
library(class)
```

```
knn.pred = knn(as.matrix(Weekly[train,"Lag2"]), as.matrix(Weekly[!train,"Lag2"]), Weekly[train,"Directi
table(Weekly[!train,]$Direction, knn.pred)
```

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

```
mean(knn.pred == Weekly[!train,]$Direction)
```

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

**(h)**

LDA has best predicition

**(i)**

Predictor: Lag2
KNN: 10

```
knn.pred = knn(as.matrix(Weekly[train,"Lag2"]), as.matrix(Weekly[!train,"Lag2"]), Weekly[train,"Directi
mean(knn.pred == Weekly[!train,]$Direction)
```

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

Predictor: Lag2
KNN: 100

```
knn.pred = knn(as.matrix(Weekly[train,"Lag2"]), as.matrix(Weekly[!train,"Lag2"]), Weekly[train,"Directi
mean(knn.pred == Weekly[!train,]$Direction)
```

```
## [1] 0.5576923
```

Predictor: Lag1 Lag2
KNN: 1

```
knn.pred = knn(as.matrix(Weekly[train, c("Lag1", "Lag2")]), as.matrix(Weekly[!train,c("Lag1", "Lag2")])
mean(knn.pred == Weekly[!train,]$Direction)
```

```
## [1] 0.4807692
```

Predictor: Lag1 Lag2, Lag3, Lag3, Lag4
KNN: 1

```
knn.pred = knn(as.matrix(Weekly[train, c("Lag1", "Lag2", "Lag3", "Lag4", "Lag5")]), as.matrix(Weekly[!t
mean(knn.pred == Weekly[!train,]$Direction)
```

```
## [1] 0.4807692
```

Predictor: Lag1 Lag2
LDA

```
lda.fit = lda(Direction ~ Lag1 + Lag2, data = Weekly, subset = train)
lda.pred = predict(lda.fit , newdata= Weekly[!train,], type="response")
mean(lda.pred$class==Weekly[!train,]$Direction)
```

```
## [1] 0.5769231
```

Predictor: Lag1 Lag2,Lag3, Lag4, Year
LDA

```
lda.fit = lda(Direction ~ Lag1+Lag2+Lag3+Lag4+Year, data = Weekly, subset = train)
lda.pred = predict(lda.fit , newdata= Weekly[!train,], type="response")
mean(lda.pred$class==Weekly[!train,]$Direction)
```

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

^^^ The best result

# Problem 6

### Chapter 4, Exercise 11

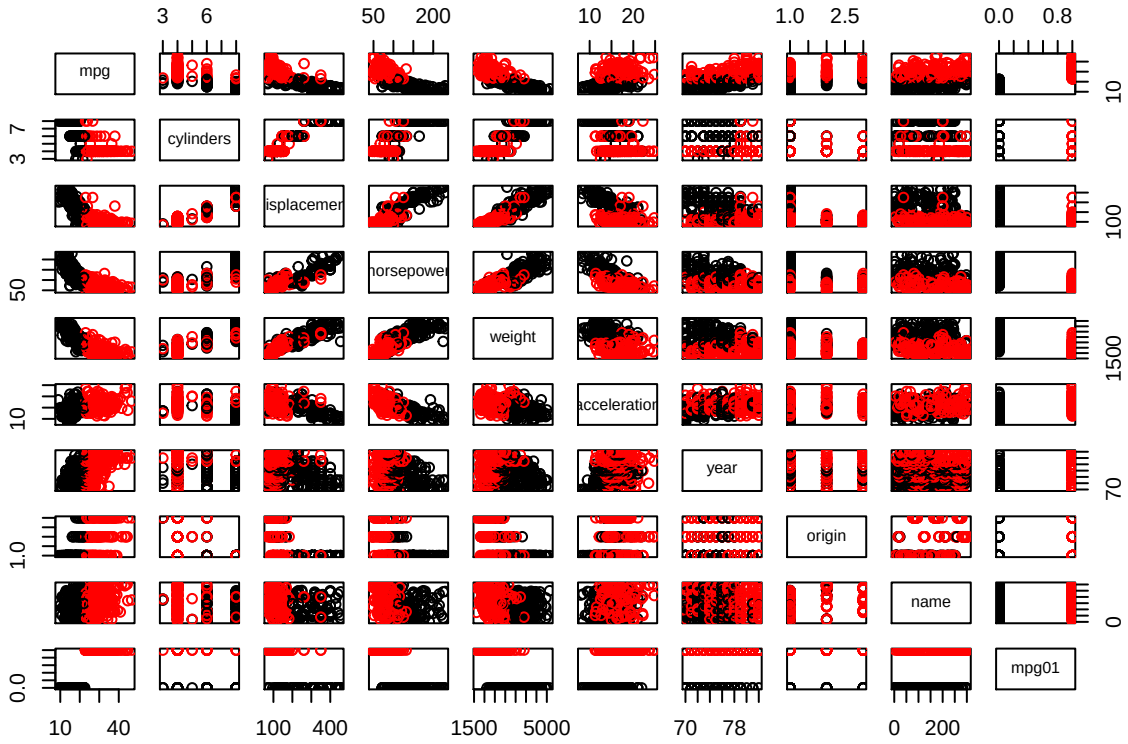
### (a)

```
mpg01 = ifelse(Auto$mpg > median(Auto$mpg), 1, 0)
Auto = data.frame(Auto, mpg01)
attach(Auto)
```

```
## The following object is masked _by_ .GlobalEnv:
##
##     mpg01
```

### (b)

```
pairs(Auto, col=Auto$mpg01+1)
```

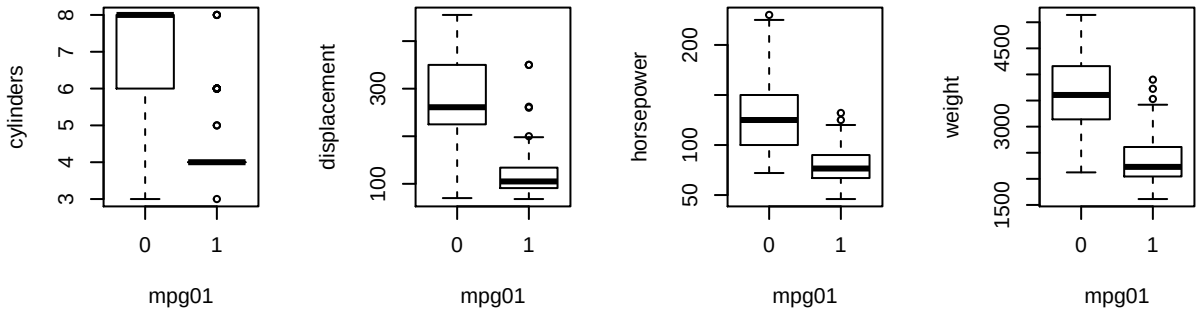


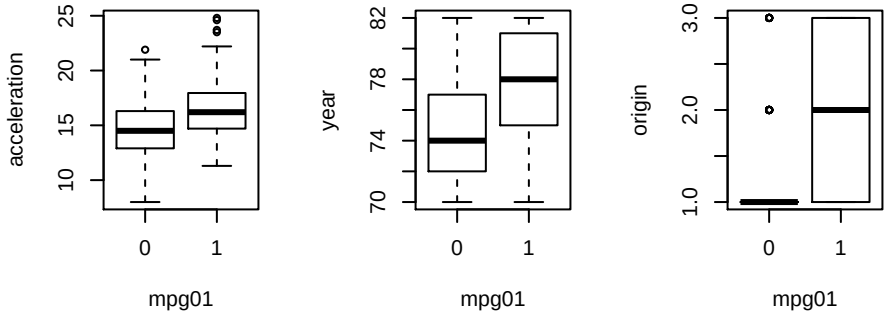
```
cor(Auto[,-9])
```

```
##                     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
```

mpg01 has positive relation with year and origin
mpg01 has opposite relation with cylinders, displacement, horsepower and weight

```
par(mfrow=c(2,4))
boxplot( cylinders ~ mpg01)
boxplot( displacement ~ mpg01)
boxplot( horsepower ~ mpg01)
boxplot( weight ~ mpg01)
boxplot( acceleration ~ mpg01)
boxplot( year ~ mpg01)
boxplot( origin ~ mpg01)
```




cylinders, displacement, horsepower and weight medinas (middle line) is outsite of other box so they are separate groups

### (c)

```
train = year%%2 == 0
print("Train:")
```

```
## [1] "Train:"
```

```
dim(Auto[train,])[1]
```

```
## [1] 210
```

```
print("Test:")
```

```
## [1] "Test:"
```

```
dim(Auto[!train,])[1]
```

```
## [1] 182
```

```
print("Total:")
```

```
## [1] "Total:"
```

```
dim(Auto)[1]
```

```
## [1] 392
```

### (d)

lda:

```
lda.fit =  lda(mpg01 ~ cylinders + displacement + horsepower + weight, data = Auto , subset = train)
lda.pred = predict(lda.fit, newdata = Auto[!train,], type="response")
table(lda.pred$class, Auto[!train, "mpg01"])
```

```
##
##      0  1
##   0 86  9
##   1 14 73
```

```
mean(lda.pred$class != Auto[!train, "mpg01"])
```

```
## [1] 0.1263736
```

### (e)

qda:

```
qda.fit =  qda(mpg01 ~ cylinders + displacement + horsepower + weight, data = Auto , subset = train)
qda.pred = predict(qda.fit, newdata = Auto[!train,], type="response")
table(qda.pred$class, Auto[!train, "mpg01"])
```

```
##
##      0  1
##   0 89 13
##   1 11 69
```

```
mean(qda.pred$class != Auto[!train, "mpg01"])
```

```
## [1] 0.1318681
```

### (f)

linear regression:

```
glm.fit = glm(mpg01 ~ cylinders + displacement + horsepower + weight, data =Auto, subset = train, family
glm.probs = predict(glm.fit, newdata = Auto[!train,], type = "response")
glm.pred = ifelse(glm.probs>0.5, 1, 0)
table(glm.pred, Auto[!train, "mpg01"])
```

```
##
## glm.pred  0  1
##        0 89 11
##        1 11 71
```

```
mean(glm.pred != Auto[!train, "mpg01"])
```

```
## [1] 0.1208791
```

^^^ lowest test error

### (g)

KNN (k=1)

```
knn.pred = knn(Auto[train,c("cylinders", "displacement", "horsepower", "weight")], Auto[!train,c("cylin
table(knn.pred =  Auto[!train, "mpg01"])
```

```
## knn.pred
##   0   1
## 100  82
```

```
mean(knn.pred != Auto[!train, "mpg01"])
```

```
## [1] 0.1538462
```

KNN (k=10)

```
knn.pred = knn(Auto[train,c("cylinders", "displacement", "horsepower", "weight")], Auto[!train,c("cylin
mean(knn.pred != Auto[!train, "mpg01"])
```

```
## [1] 0.1648352
```

KNN (k=100)

```
knn.pred = knn(Auto[train,c("cylinders", "displacement", "horsepower", "weight")], Auto[!train,c("cylin
mean(knn.pred != Auto[!train, "mpg01"])
```

```
## [1] 0.1428571
```

^^^ best KNN (k=100)