Problem 1

(a) 10%

(b) $10\% \times 10\% = 1\%$

(c) 10^{-100}

(d) For a fixed p and sample size N, if we want to use training observations that are "near" to a given test sample (near means the training observations are with f range of the test sample in every dimention), then we can only have $N \cdot f^{-p}$ training observations. f^{-p} is very small for large p as $0<f<1$. this means we will have very few training observations 'near' any given test sample.

(e) $p=1$: 0.1

$p=2$: $\sqrt{0.1} = 0.316$

$p=100$: $0.1^{\frac{1}{100}} = 0.977$

The answer shows that if we want to use a fixed 'reasonably large' fraction of training observations to make prediction for any give test sample, we will have to use traing observations that are further and further from the test sample in each dimension. These traing observations are no longer "local" to the test sample and the KNN method is expected to perform poorly.

Problem 2

(a) On the training set, QDA will be better. Because it is more flexible. In fact, QDA is fitting the model for normal distributions that do not necessarily have the same covariance matrices, which contains the situation where all the normals have the same covariance matrices as a subset. Therefore, QDA will fit the training data better.

On the test set, LDA will out perform. Because QDA might over fit the training data, leading to high variance in the $\hat{P}(Y|X)$. However, if n is large, the two model could have similar results.

(b) On the training set, QDA still is better, for some reason as (a).

On the test set, there's not enough information to tell. If the true decision boundary is more close to linear, LDA may out perform, whereas if the true boundary is more close to quadratic, QDA may out perform.

(c) Improve. Because higher n will lead to lower variance for both models. This makes the disadvantage of high variance for QDA to decline gradually. But QDA has lower bias than LDA, therefore, QDA will become better as n increases

(d) True for big training size n. Because if n is big, QDA is less likely to overfit the training data, so it will have an almost linear decision boundary. This makes the test error for LDA and QDA to be similar, then for a certain test set, chances are that QDA will perform better

But for n is small, QDA will overfit the training data and perform worse than LDA

Problem 3.

(a) $\hat{p}(Y=1 \mid X_1=40, X_2=35) = \frac{1}{1+e^{-(-6+0.05\times 40+1\times 35)}} = 0.3775$

(b) $0.5 = \frac{1}{1+e^{-(-6+0.05x+35)}}$

$e^{2.5-0.05x} = 1$.

$2.5-0.05x = 0$

$x = 50$

∴ the student needs to study 50 hs

Problem 4

We should use logistic regression. Because the 1-NN method would make no mistake on the training data. Suppose the training and test data both have size N, then the test error rate for 1-NN is: $\frac{18\% \times 2N}{N} = 36\%$, which is higher that the test error rate for logistic regression. So we should use logistic regression.

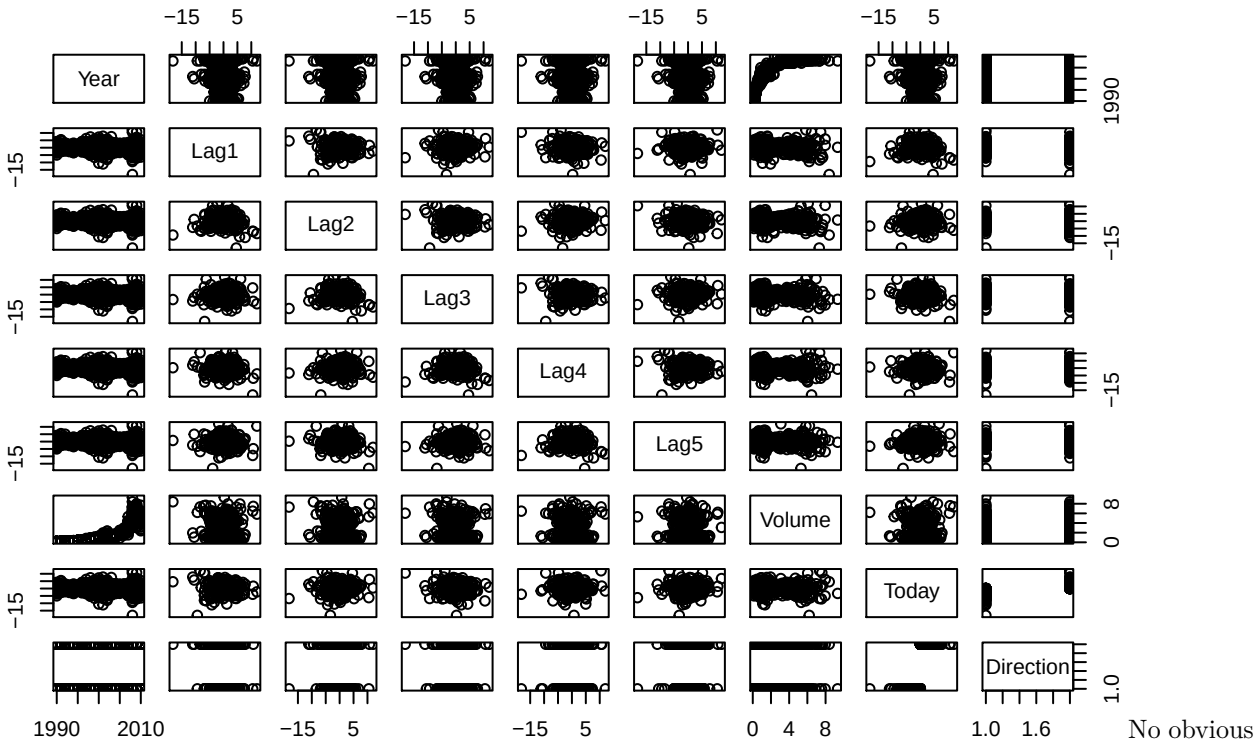
Problem 5

(a)

```
library(ISLR)
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
```

```
plot(Weekly)
```



No obvious

pattern can be discovered except that volume becomes bigger year by year.

(b)

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

```
##
## 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 is statistically significant

(c)

```
logit.prob=predict(logit.out,type='response')
logit.predict=rep('Down',length(Direction))
logit.predict[logit.prob>0.5]='Up'
table(logit.predict,Direction)
```

```
##              Direction
## logit.predict Down  Up
##          Down   54  48
##          Up    430 557
```

If predicting 'Up' is positive, then the logistic has a false positive rate of $\frac{430}{430+54} = 0.888$ and a false negative rate of $\frac{48}{48+557} = 0.079$

(d)

```
training=Weekly[Year>=1990 & Year<=2008,]
test=Weekly[Year<1990 | Year>2008,]
logit.out2=glm(Direction~Lag2,data=training,family=binomial)
logit.predict2=predict(logit.out2,newdata=test,type='response')
logit.prob2=rep('Down',dim(test)[1])
logit.prob2[logit.predict2>0.5]='Up'
table(logit.prob2,test$Direction)
```

```
##
## logit.prob2 Down Up
##        Down    9  5
##        Up     34 56
```

```
(9+56)/(9+5+34+56)
```

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

The overall fraction of correct predictions for the held out data is 62.5%

(e)

```
library(MASS)
lda.out=lda(Direction~Lag2,data=training)
lda.predict=predict(lda.out,newdata=test)
table(lda.predict$class,test$Direction)
```

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

```
(9+56)/(9+5+34+56)
```

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

The overall fraction of correct predictions for the held out data is 62.5%

(f)

```
qda.out=qda(Direction~Lag2,data=training)
qda.predict=predict(qda.out,newdata=test)
table(qda.predict$class,test$Direction)
```

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

```
(61)/(61+43)
```

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

The overall fraction of correct predictions for the held out data is 58.65%

(g)

```
library(class)
knn.out=knn(data.frame(scale(training$Lag2)),data.frame(scale(test$Lag2)),training$Direction,k=1)
table(knn.out,test$Direction)
```

```
##
## knn.out Down Up
##    Down   15 26
##    Up     28 35
```

```
(14+36)/(14+25+29+36)
```

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

The overall fraction of correct predictions for the held out data is 48.08%

(h) LDA and Logistic regression

(i) We begin with the null model: Direction~Lag2 and try to add interactions between Lag2 and other predictors one at a time.

```
lda.fit = lda(Direction ~ Lag2+Lag2:Lag1, data = training)
lda.pred = predict(lda.fit, newdata = test)
mean(lda.pred$class == test$Direction)
```

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

```
lda.fit = lda(Direction ~ Lag2+Lag2:Lag3, data = training)
lda.pred = predict(lda.fit, newdata = test)
mean(lda.pred$class == test$Direction)
```

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

```
lda.fit = lda(Direction ~ Lag2+Lag2:Lag4, data = training)
lda.pred = predict(lda.fit, newdata = test)
mean(lda.pred$class == test$Direction)
```

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

```
lda.fit = lda(Direction ~ Lag2+Lag2:Lag5, data = training)
lda.pred = predict(lda.fit, newdata = test)
mean(lda.pred$class == test$Direction)
```

```
## [1] 0.6346154
```

```
lda.fit = lda(Direction ~ Lag2+Lag2:Year, data = training)
lda.pred = predict(lda.fit, newdata = test)
mean(lda.pred$class == test$Direction)
```

```
## [1] 0.6153846
```

```
lda.fit = lda(Direction ~ Lag2+Lag2:Volume, data = training)
lda.pred = predict(lda.fit, newdata = test)
mean(lda.pred$class == test$Direction)
```

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

According to the result, it seems that we will add the interaction of Lag2 and Lag5 to the null model Direction~Lag2. Then we examine if there is any other interactions that is worth adding.

```
lda.fit = lda(Direction ~ Lag2+Lag2:Lag5+Lag2:Lag1, data = training)
lda.pred = predict(lda.fit, newdata = test)
mean(lda.pred$class == test$Direction)
```

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

```
lda.fit = lda(Direction ~ Lag2+Lag2:Lag1+Lag2:Lag3, data = training)
lda.pred = predict(lda.fit, newdata = test)
mean(lda.pred$class == test$Direction)
```

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

```
lda.fit = lda(Direction ~ Lag2+Lag2:Lag1+Lag2:Volume, data = training)
lda.pred = predict(lda.fit, newdata = test)
mean(lda.pred$class == test$Direction)
```

```
## [1] 0.6153846
```

```
lda.fit = lda(Direction ~ Lag2+Lag2:Lag1+Lag2:Year, data = training)
lda.pred = predict(lda.fit, newdata = test)
mean(lda.pred$class == test$Direction)
```

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

```
lda.fit = lda(Direction ~ Lag2+Lag2:Lag1+Lag2:Lag3, data = training)
lda.pred = predict(lda.fit, newdata = test)
mean(lda.pred$class == test$Direction)
```

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

As can be seen, none of those models above can further decrease the test error, so we stop selection and end up having the model: Direction ~ Lag2 + Lag2:Lag5

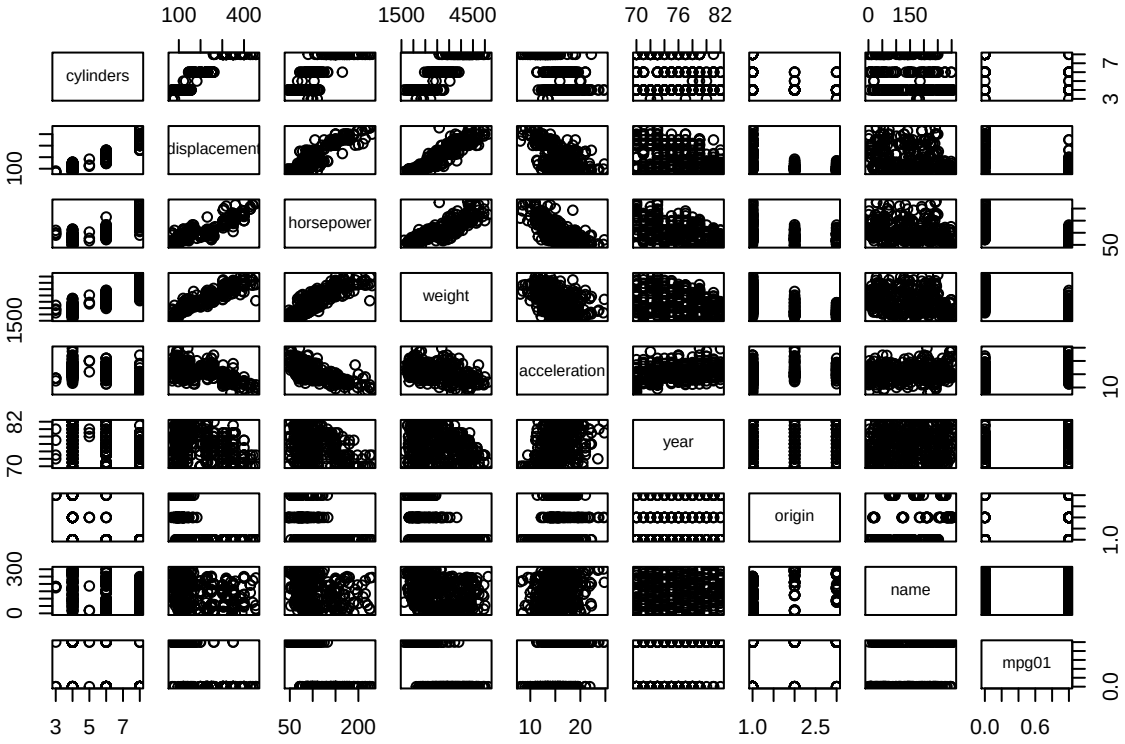
Problem 6

(a)

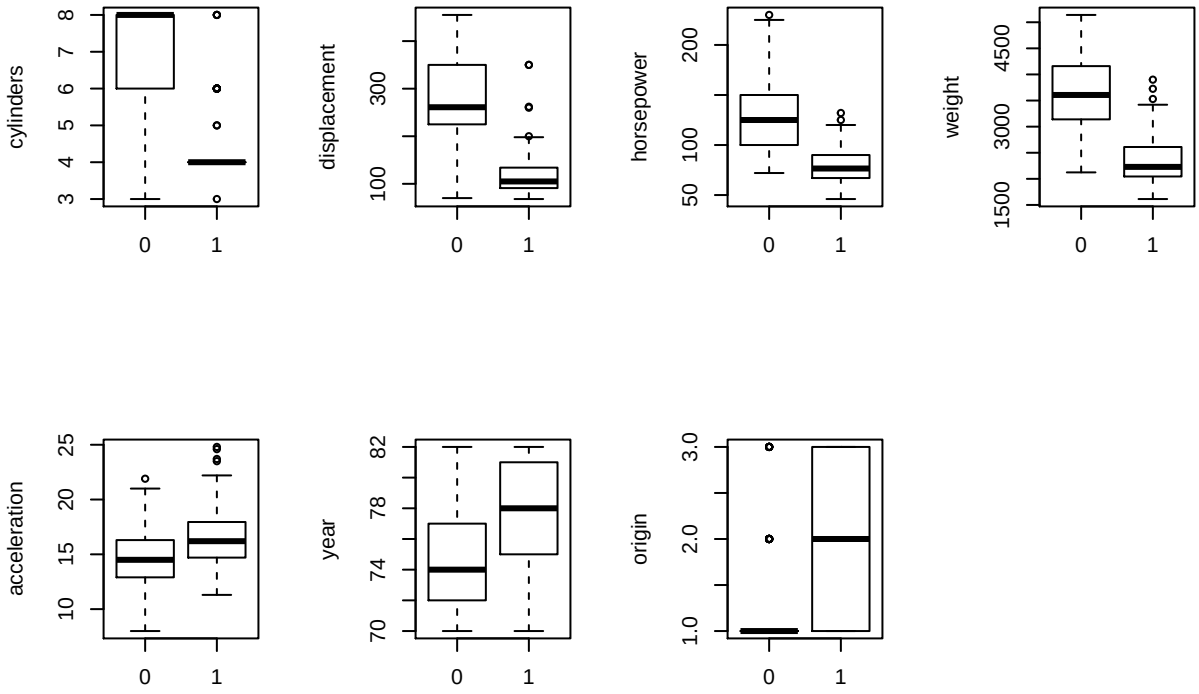
```
attach(Auto)
mpg01=rep(0,length(mpg))
mpg01[mpg>median(mpg)]=1
mydata=data.frame(Auto[,-1],mpg01)
```

(b)

```
plot(mydata)
```




```
n=names(mydata)
par(mfrow=c(2,4))
for(i in 1:7)
{
  boxplot(mydata[,i]~mpg01,ylab=n[i])
}
```



displacement,horsepower,weight,acceleration seems to be most useful

(c)

```
set.seed(1)
rsample=sample(1:length(mpg01),size=length(mpg01)/2)
train=mydata[rsample,]
test=mydata[-rsample,]
```

(d)

```
lda.fit=lda(mpg01~displacement+horsepower+weight+acceleration,data=train)
lda.predict=predict(lda.fit,newdata=test)
table(lda.predict$class,test$mpg01)
```

```
##
##        0   1
##    0  73   0
##    1  20 103
```

```
(7+15)/(85+7+15+89)
```

```
## [1] 0.1122449
```

The model's test error rate is 11.22%

(e)

```
qda.fit=qda(mpg01~displacement+horsepower+weight+acceleration,data=train)
qda.predict=predict(qda.fit,newdata=test)
table(qda.predict$class,test$mpg01)
```

```
##
##      0  1
##   0 75  6
##   1 18 97
```

```
(7+10)/(90+7+10+89)
```

```
## [1] 0.08673469
```

The model's test error rate is 8.67%

(f)

```
logit.fit=glm(mpg01~displacement+horsepower+weight+acceleration,data=train,family=binomial)
logit.prob=predict(logit.fit,newdata=test,type='response')
logit.predict=rep(0,length(test$mpg01))
logit.predict[logit.prob>0.5]=1
table(logit.predict,test$mpg01)
```

```
##
## logit.predict  0  1
##             0 76  6
##             1 17 97
```

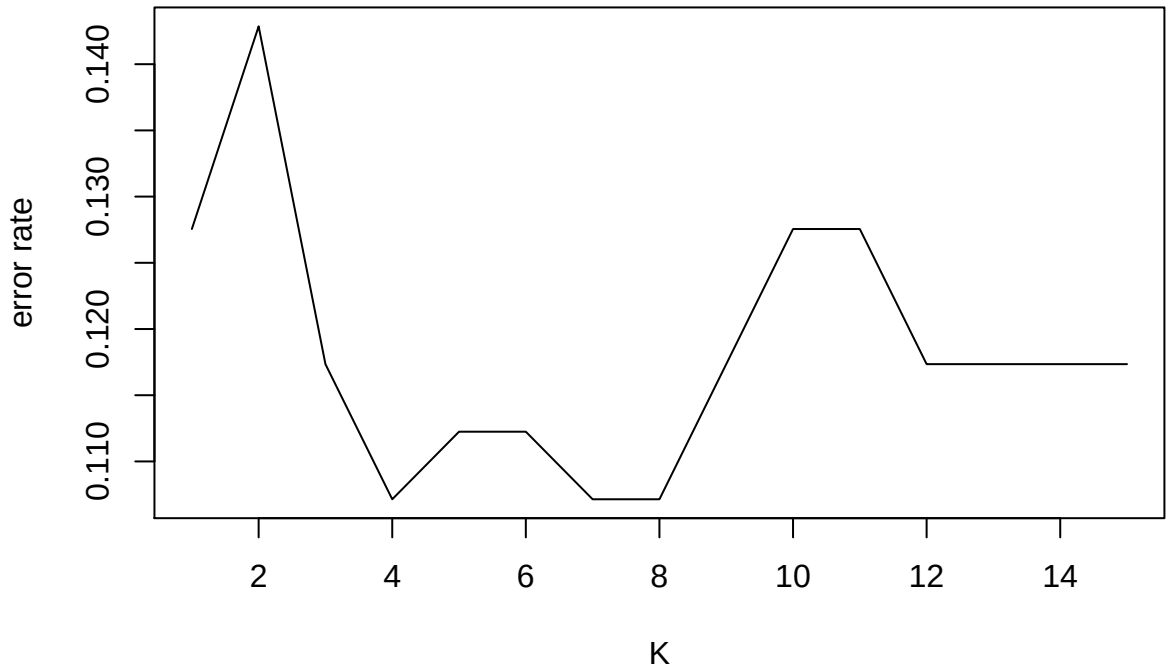
```
(12+12)/(88+12+12+84)
```

```
## [1] 0.122449
```

The model's test error rate is 12.24%

(g)

```
vec_error_rate=rep(0,15)
par(mfrow=c(1,1))
for (k in 1:15){
knn.fit=knn(train[2:5],test[2:5],train$mpg01,k)
vec_error_rate[k]=sum(knn.fit!=test$mpg01)/length(test$mpg01)
}
plot(vec_error_rate,xlab='K',ylab='error rate',main='Error rate V.S. the value of K',type='l')
```



$K = 6$ or $K = 7$ seems to perform the best on this data set because they have good result and they