HW2 Classification

2/1/2022

4.6

(a)

```
(p <- invlogit(-6+0.05*40+1*3.5))
```

Answer:

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

(b)

```
(hour <- (logit(0.5)+6-1*3.5)/0.05)
```

Answer:

```
## [1] 50
```

4.8

Answer: When we use 1-nearest neighbors, the error rate on training set is 0. So, the error rate of 1-NN on test set is 18%*2 = 36%, is bigger than the error rate of logistic classifier on testing set, which is 30%.

The logistic classification is better.

4.9

(a)

```
(p <- 0.37/1.37)
```

Answer:

```
## [1] 0.270073
```

(b)

```
(odd <- 0.16/(1-0.16))
```

Answer:

```
## [1] 0.1904762
```

4.13

(a)

```
library(ISLR2)
```

Answer:

```
## Warning: package 'ISLR2' was built under R version 4.1.2
```

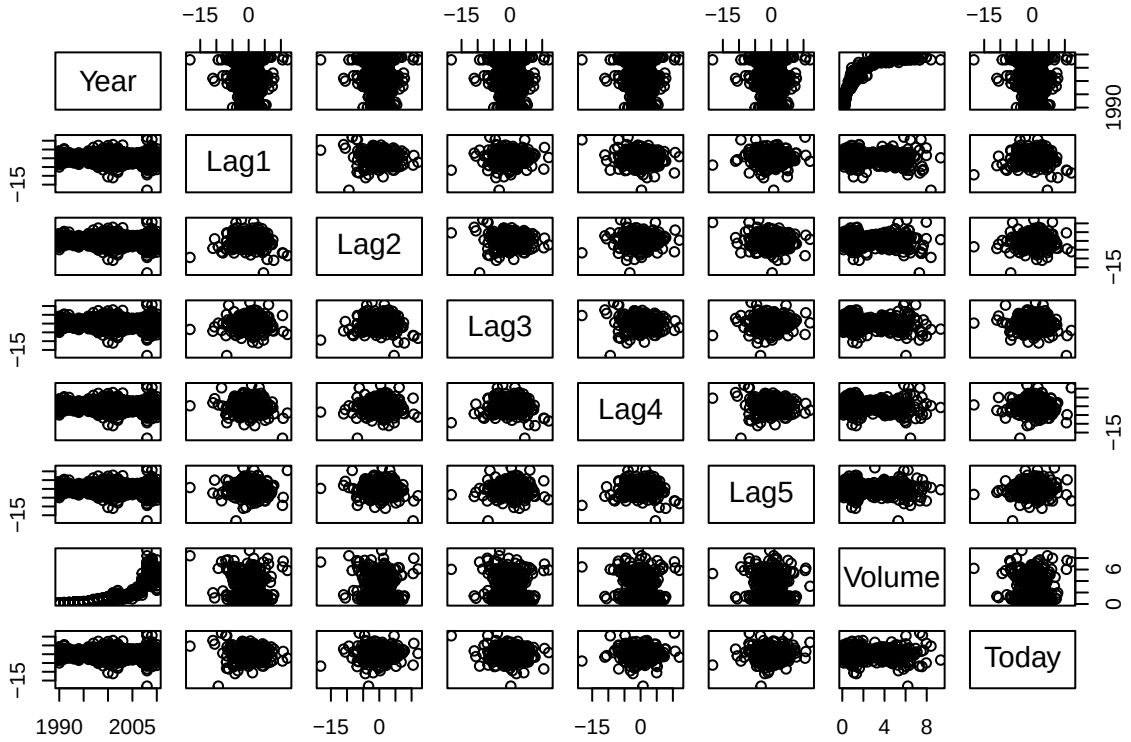
```
## numeric summary
summary(Weekly)
```

```
##       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            Today         
##  Min.   :-18.1950   Min.   :-18.1950   Min.   :0.08747   Min.   :-18.1950  
##  1st Qu.: -1.1580   1st Qu.: -1.1660   1st Qu.:0.33202   1st Qu.: -1.1540  
##  Median :  0.2380   Median :  0.2340   Median :1.00268   Median :  0.2410  
##  Mean   :  0.1458   Mean   :  0.1399   Mean   :1.57462   Mean   :  0.1499  
##  3rd Qu.:  1.4090   3rd Qu.:  1.4050   3rd Qu.:2.05373   3rd Qu.:  1.4050  
##  Max.   : 12.0260   Max.   : 12.0260   Max.   :9.32821   Max.   : 12.0260  
##  Direction 
##  Down:484  
##  Up  :605  
##            
##            
##            
##
```

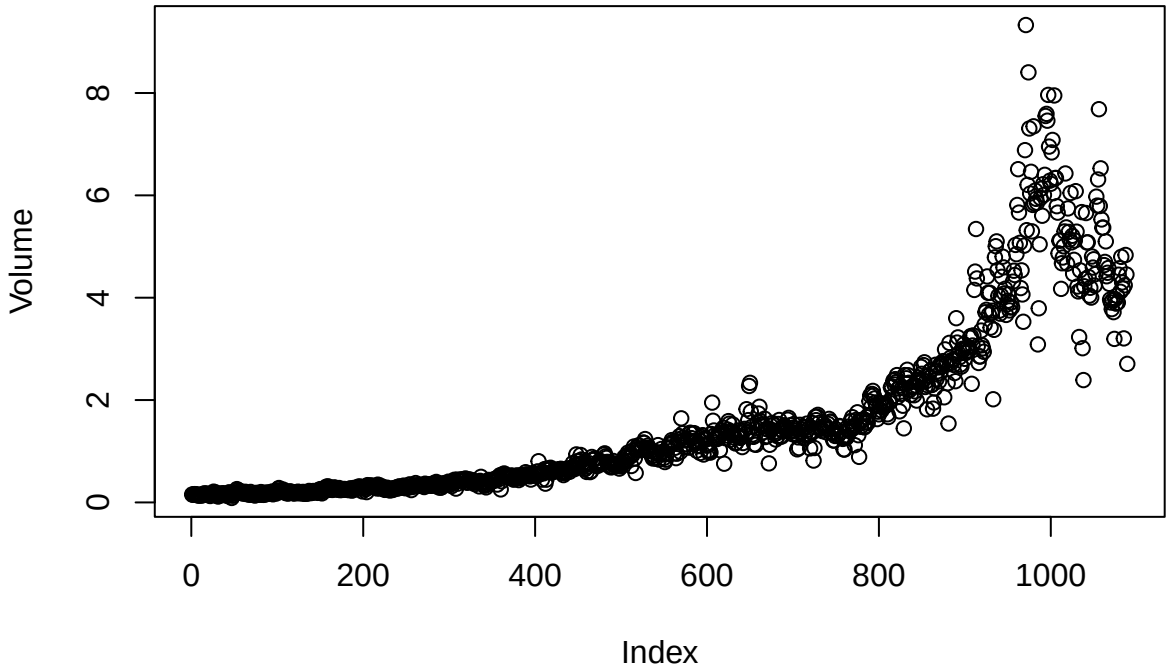
```
cor(Weekly[,-9])
```

```
##                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
```

```
## graphic summary
pairs(Weekly[,-9])
```




```
attach(Weekly) #for convenience
plot(Volume)
```



From the summaries of Weekly data, we can see the correlations between today's returns and previous days' returns are small, the only substantial correlation lies on year and volume, what's more, from the plot we can see the volume is increasing over time.

(b)

```
names(Weekly)
```

Answer:

```
## [1] "Year"      "Lag1"      "Lag2"      "Lag3"      "Lag4"      "Lag5"
## [7] "Volume"    "Today"     "Direction"
```

```
model13.1 <- glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume, family = binomial(link = "logit"), data=Weel
summary(model13.1)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
##     Volume, family = binomial(link = "logit"), 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
```

Only coefficient of Lag2 shows significance.

(c)

```
predict13.1 <- predict(model13.1, type = "response")

predglm <- rep("Down",length(predict13.1))
predglm[predict13.1>0.5] <- "Up"   ###
##confusion matrix
table(predglm, Direction)
```

Answer:

```
##        Direction
## predglm Down  Up
##    Down   54  48
```

```
##    Up    430 557
```

```
##correct prediction
mean(predglm == Direction)
```

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

The overall percentage of correct prediction is 56%. When market goes up, the true positive rate is 557/(557+48)=92%, when market goes down, the true negative rate is 54/(54+430)=11.1%. So the logistic model predicts more precisely when market goes up.

(d)

```
##test our model in test set

# traindata <- Weekly[Year<=2008,]
# testdata <- Weekly[Year>2008,]
train <- (Year<2009) #Boolean vector, true or false
Weekly0910 <- Weekly[!train,]

model13.2 <- glm(Direction~Lag2,family = binomial(link = "logit"), data=Weekly, subset = train)

predict13.2 <- predict(model13.2, Weekly0910, type = "response")
predglm2 <- rep("Down",length(predict13.2))
predglm2[predict13.2>0.5] <- "Up"
Direction0910 <- Direction[!train]
##confusion matrix
table(predglm2, Direction0910)
```

Answer:

```
##         Direction0910
## predglm2 Down Up
##     Down    9  5
##     Up     34 56
```

```
##correct prediction
mean(predglm2 == Direction0910)
```

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

The overall percentage of correct prediction is 62.5%. When market goes up, the true positive rate is 56/(56+5)=91.8%, when market goes down, the true negative rate is 9/(9+34)=20.9%. So the logistic model predicts more precisely when market goes up.

(e)

```
##LDA
library(MASS)
```

Answer:

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:ISLR2':
##
```

```
##     Boston
```

```
fit.lda <- lda(Direction~Lag2, data=Weekly, subset = train)
#(fit.lda)
predlda <- predict(fit.lda, Weekly0910)
#confusion matrix
table(predlda$class, Direction0910)
```

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

```
##correct prediction
mean(predlda$class == Direction0910)
```

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

The prediction by LDA is exactlly same with logistic classification.

(f)

```
##QDA
fit.qda <- qda(Direction~Lag2, data=Weekly, subset = train)

predqda <- predict(fit.qda, Weekly0910)
#confusion matrix
table(predqda$class, Direction0910)
```

Answer:

```
##       Direction0910
##        Down Up
##   Down    0  0
##   Up     43 61
```

```
##correct prediction
mean(predqda$class == Direction0910)
```

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

The overall percentage of correct prediction is 58.6%. The variance of QDA is bigger than LDA.

(g)

```
## KNN, K=1
library(class)
train.x <- as.matrix(Lag2[train])
test.x <- as.matrix(Lag2[!train])
train.Direction <- (Direction[train])

set.seed(1)
predknn <- knn(train.x, test.x, train.Direction, k=1)
#confusion matrix
table(predknn, Direction0910)
```

Answer:

```
##        Direction0910
## predknn Down Up
##    Down   21 30
##    Up     22 31
```

```
##correct prediction
mean(predknn == Direction0910)
```

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

The overall percentage of correct prediction is 50%. The output is not very good.

(h)

```
library(e1071)
fit.nb <- naiveBayes(Direction~Lag2, data = Weekly, subset = train)
prednb <- predict(fit.nb, Weekly0910)
table(prednb, Direction0910)
```

Answer:

```
##       Direction0910
## prednb Down Up
##   Down    0  0
##   Up     43 61
```

```
mean(prednb == Direction0910)
```

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

The overall percentage of correct prediction is 58.65%.

(i)

Answer: Logistic classification and LDA show best results on this data set.

(j)

```
##Logistic
logit <- glm(Direction~Lag2+Lag3,family = binomial(link = "logit"), data=Weekly, subset = train)
logitpred <- predict(logit, Weekly0910, type = "response")
predlogit <- rep("Down",length(logitpred))
predlogit[logitpred>0.5] <- "Up"
mean(predlogit == Direction0910)
```

Answer:

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

```
##LDA
lda <- lda(Direction~Lag2+Lag3, data = Weekly, subset = train)
ldapred <- predict(lda, Weekly0910)
mean(ldapred$class == Direction0910)
```

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

```
##QDA
qda <- qda(Direction~Lag1+Lag3, data = Weekly, subset = train)
```

```
qdapred <- predict(qda, Weekly0910)
mean(qdapred$class == Direction0910)
```

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

```
##KNN
set.seed(1)
knnpred <- knn(train.x, test.x, train.Direction, k = 4)
mean(knnpred == Direction0910)
```

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

```
##naive bayes
nb <- naiveBayes(Direction~Lag2+Lag3, data = Weekly, subset = train)
nbpred <- predict(nb, Weekly0910)
mean(nbpred == Direction0910)
```

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

For logistic and LDA, best predictors are lag2 and lag3

For QDA, best predictors are lag1 and lag3

For KNN, best k is 4

For naive bayes, best predictors are lag2 and lag3

4.14

(a)

```
library(ISLR2)
#summary(Auto)
attach(Auto)
```

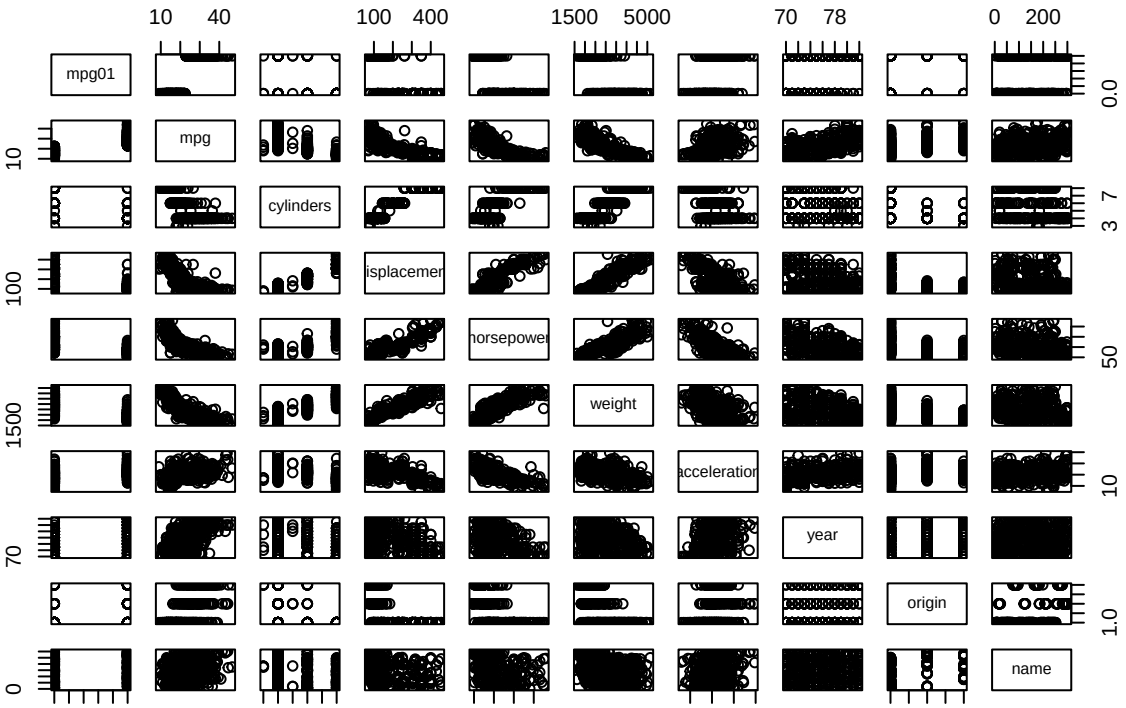
Answer:

```
## The following object is masked from package:ggplot2:
##
##     mpg
```

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

(b)

```
##scatter plot
pairs(data)
```

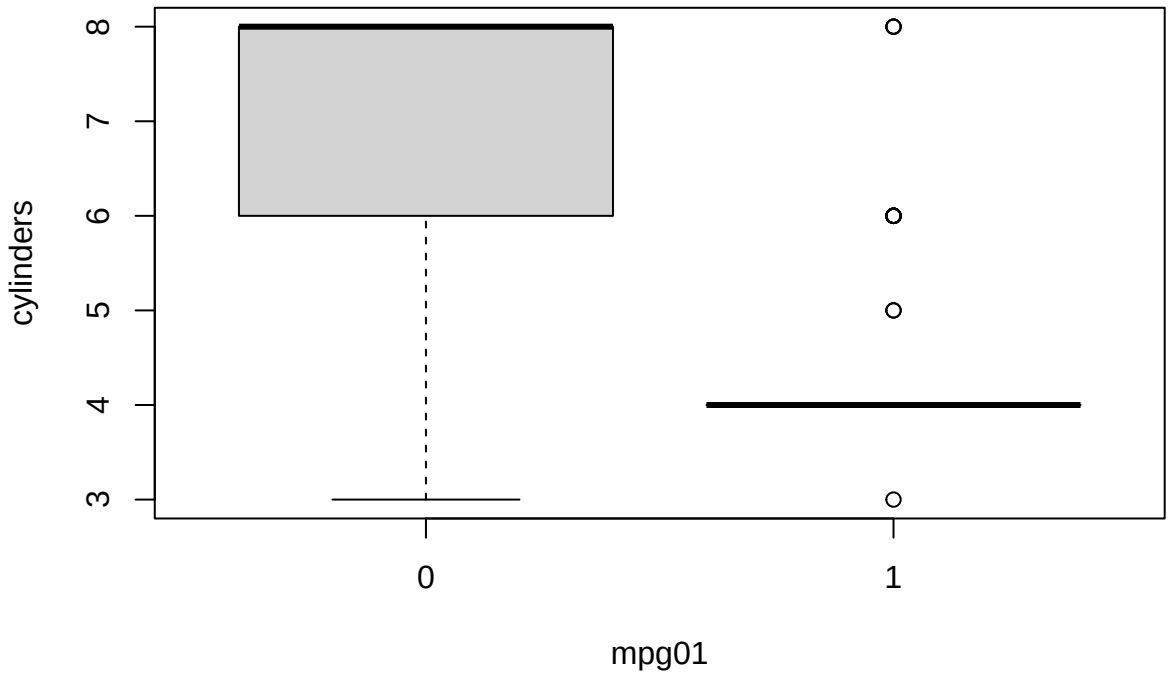


Answer:

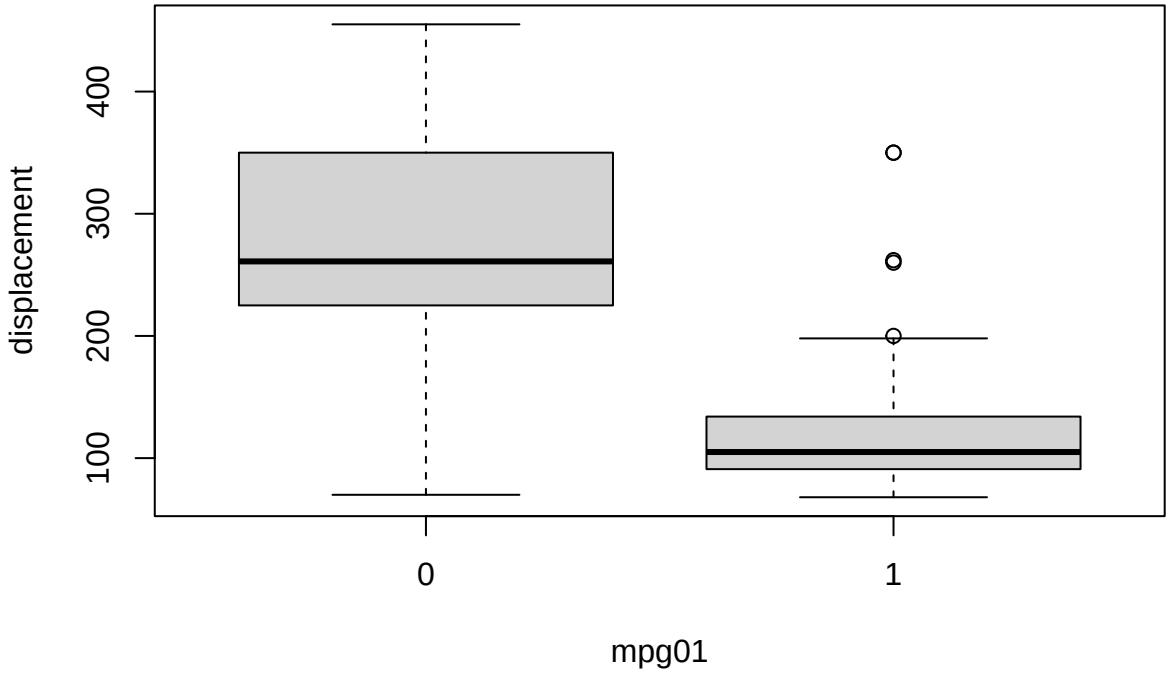
```
cor(data[,-10])
```

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

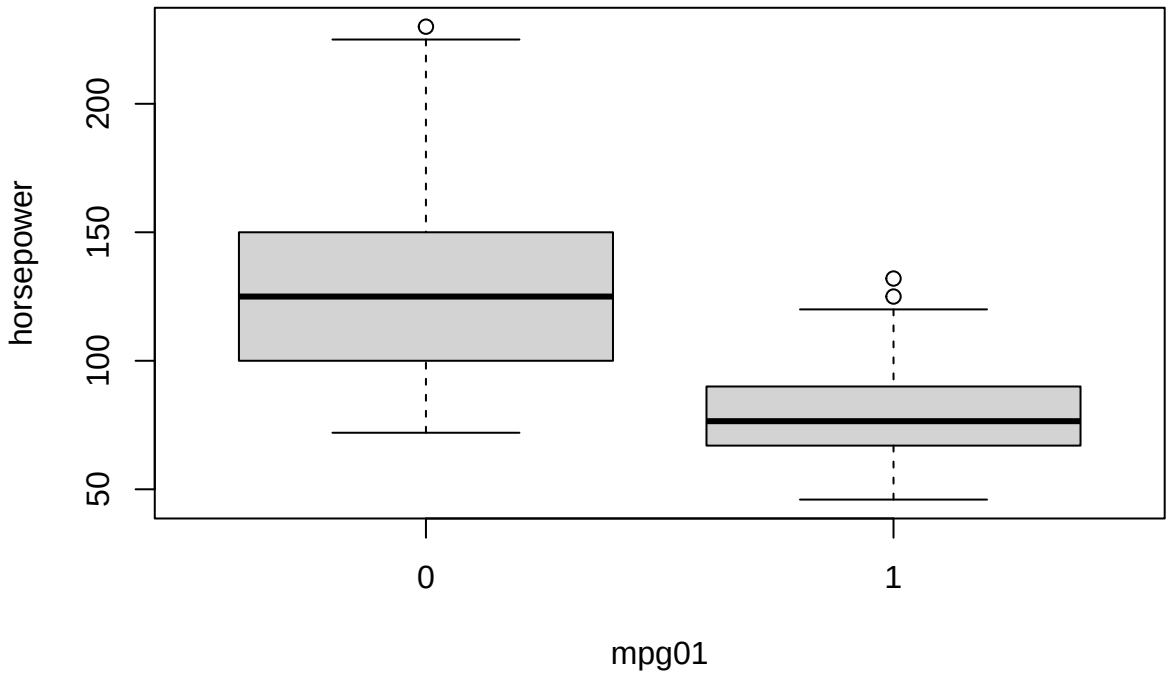
```
##boxplot
boxplot(cylinders~mpg01)
```



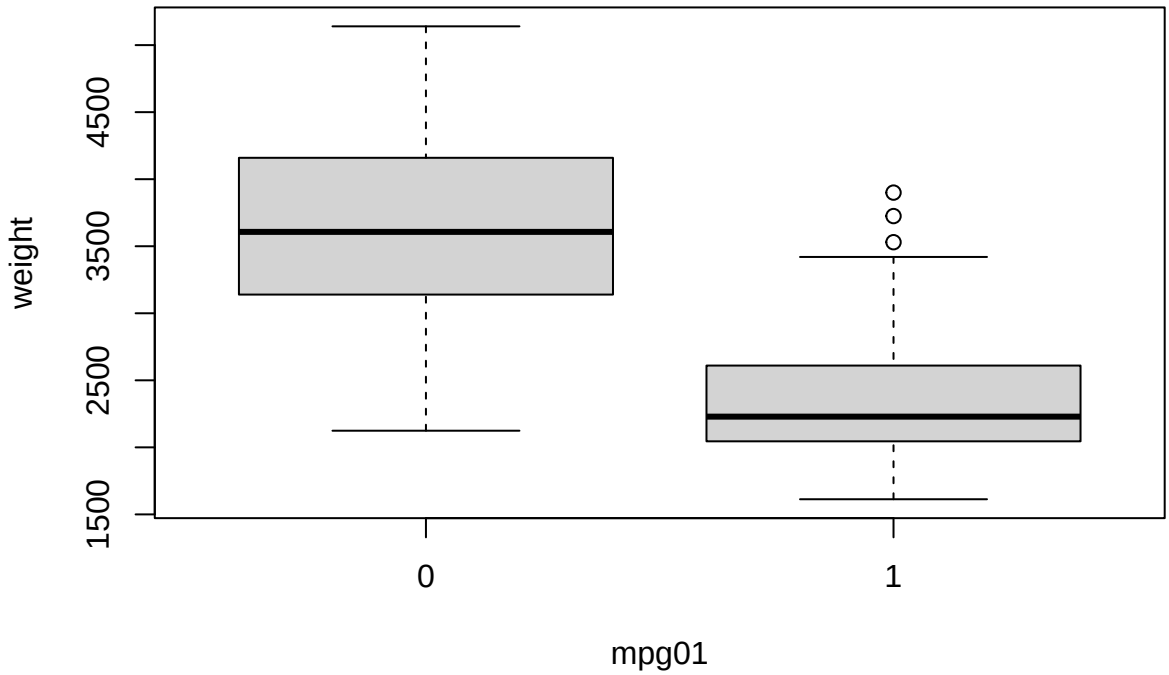

```
boxplot(displacement~mpg01)
```



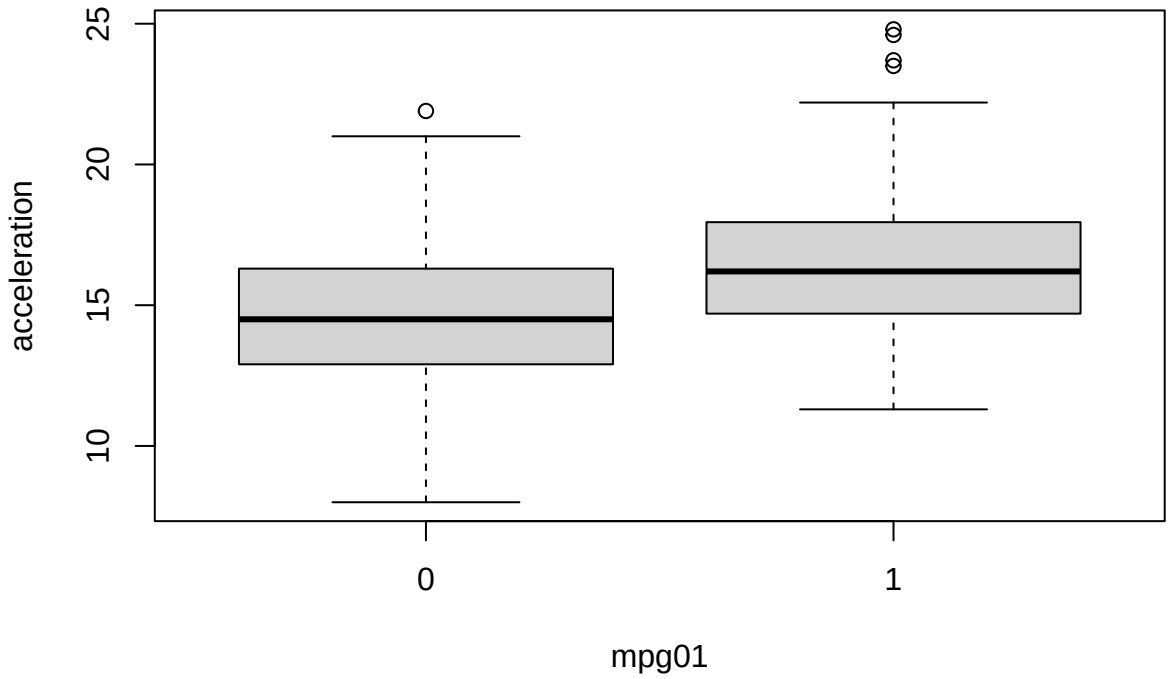

```
boxplot(horsepower~mpg01)
```



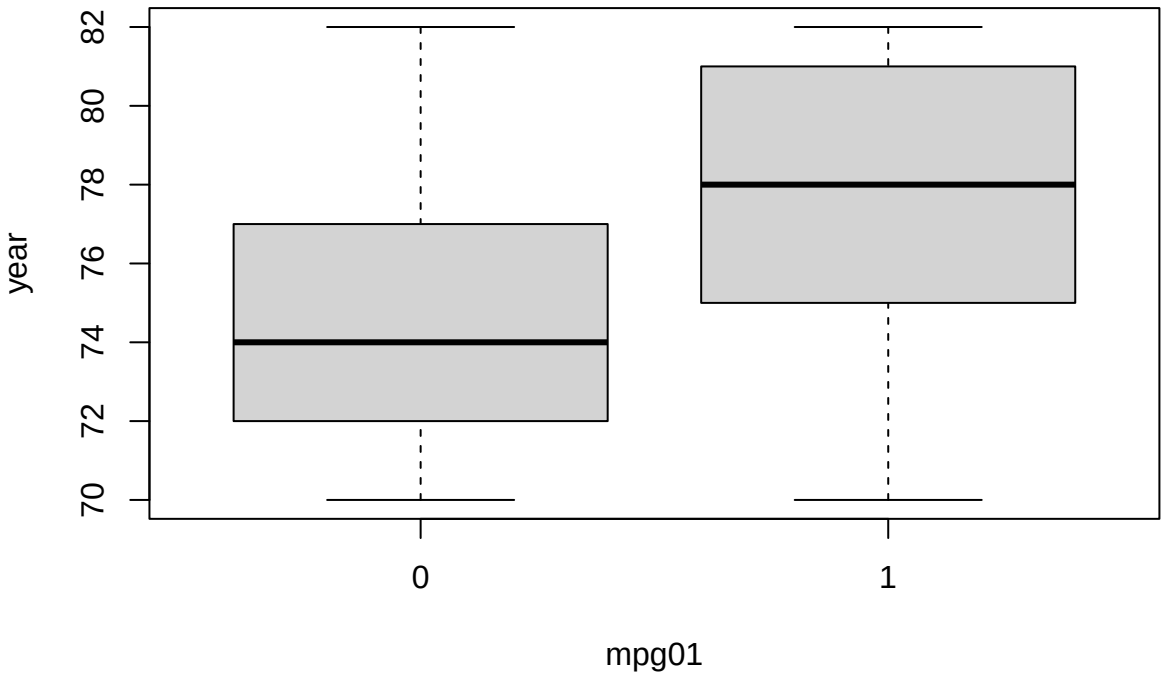

```
boxplot(weight~mpg01)
```



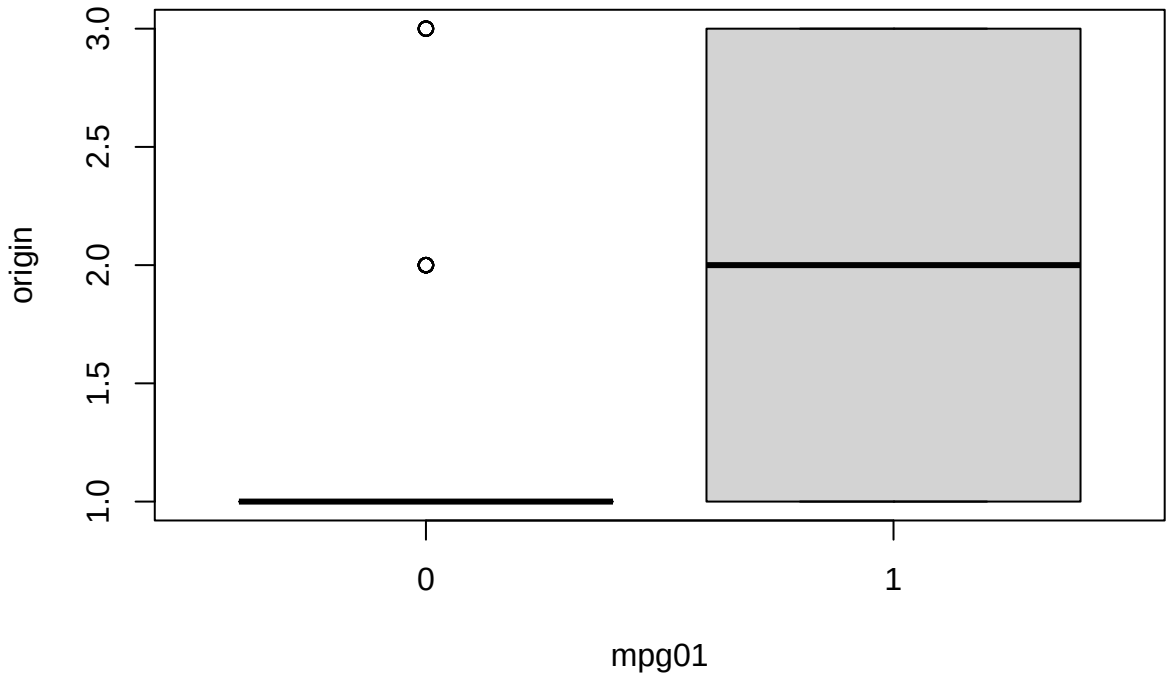

```
boxplot(acceleration~mpg01) #small
```




```
boxplot(year~mpg01)
```




```
boxplot(origin~mpg01)
```



cylinders, displacement, horsepower, weight.

(c)

```
train <- (year<79)
data.train <- data[train,]
data.test <- data[!train,]

mpg01.test <- mpg01[!train]
```

Answer:

(d) LDA

```
fit14.lda <- lda(mpg01~cylinders+displacement+horsepower+weight, data = data.train)
pred14.lda <- predict(fit14.lda, data.test)
mean(pred14.lda$class != mpg01.test)
```

Answer:

```
## [1] 0.1491228
```

The test error is 14.9%.

(e) QDA

```
fit14.qda <- qda(mpg01~cylinders+displacement+horsepower+weight, data = data.train)
pred14.qda <- predict(fit14.qda, data.test)
mean(pred14.qda$class != mpg01.test)
```

Answer:

```
## [1] 0.1754386
```

The test error is 17.5%.

(f) Logistic

```
fit14.logit <- glm(mpg01~cylinders+displacement+horsepower+weight, family=binomial, data = data.train)
pred14.prob <- predict(fit14.logit, data.test)
pred14.logit <- rep(0, length(pred14.prob))
pred14.logit[pred14.prob>0.5] <- 1
mean(pred14.logit != mpg01.test)
```

Answer:

```
## [1] 0.2719298
```

The test error is 27.2%.

(g) Naive Bayes

```
fit14.nb <- naiveBayes(mpg01~cylinders+displacement+horsepower+weight, data = data.train)
pred14.nb <- predict(fit14.nb, data.test)
mean(pred14.nb != mpg01.test)
```

Answer:

```
## [1] 0.1315789
```

The test error is 13.16%.

(h) KNN

```
train14.x <- cbind(cylinders,displacement,horsepower,weight)[train,]
test14.x <- cbind(cylinders,displacement,horsepower,weight)[!train,]
mpg01.train <- mpg01[train]

set.seed(1)
pred14.knn <- knn(train14.x, test14.x, mpg01.train, k=30)
mean(pred14.knn != mpg01.test)
```

Answer:

```
## [1] 0.1754386
```

The best K is 30, test error is 17.5%.

4.15

(a)

```
power <- function(){print(2^3)}
power()
```

Answer:

```
## [1] 8
```

(b)

```
power2 <- function(x,a){print(x^a)}
power2(2,3)
```

Answer:

```
## [1] 8
```

(c)

```
power2(10,3)
```

Answer:

```
## [1] 1000
```

```
power2(8,17)
```

```
## [1] 2.2518e+15
```

```
power2(131,3)
```

```
## [1] 2248091
```

(d)

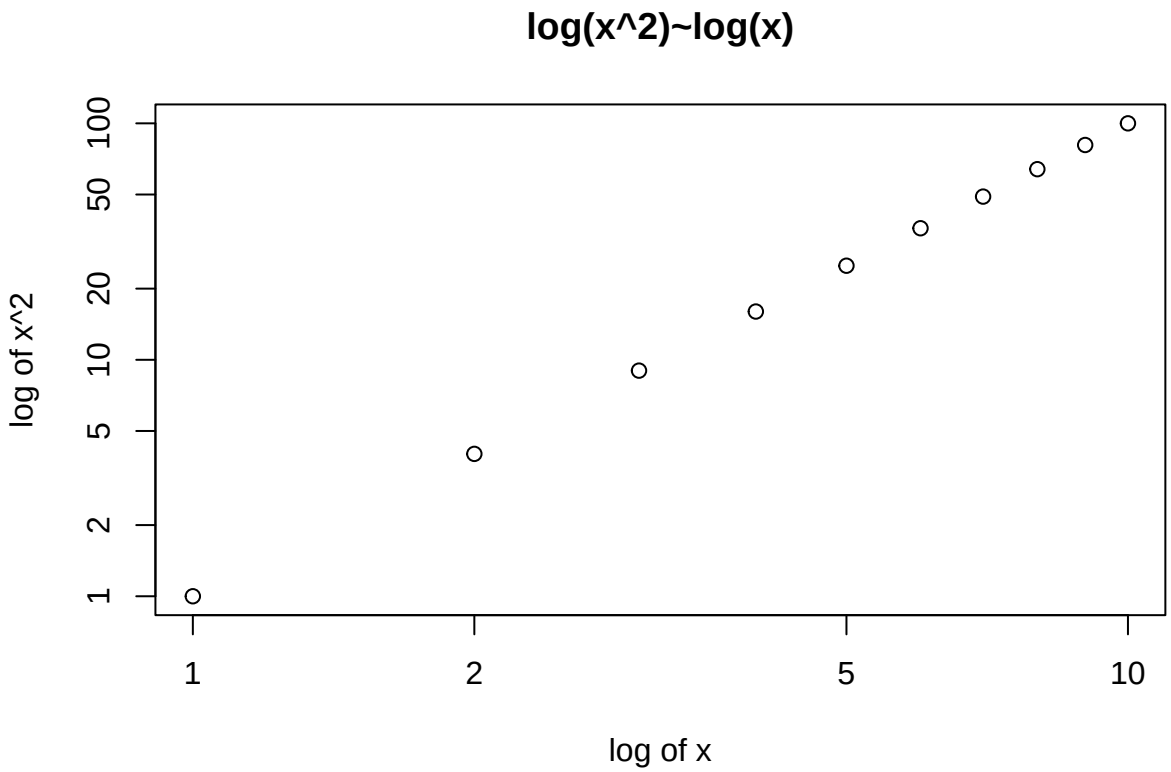
```
power3 <- function(x,a){
  result <- x^a
  return(result)
  }
power3(2,3)
```

Answer:

```
## [1] 8
```

(e)

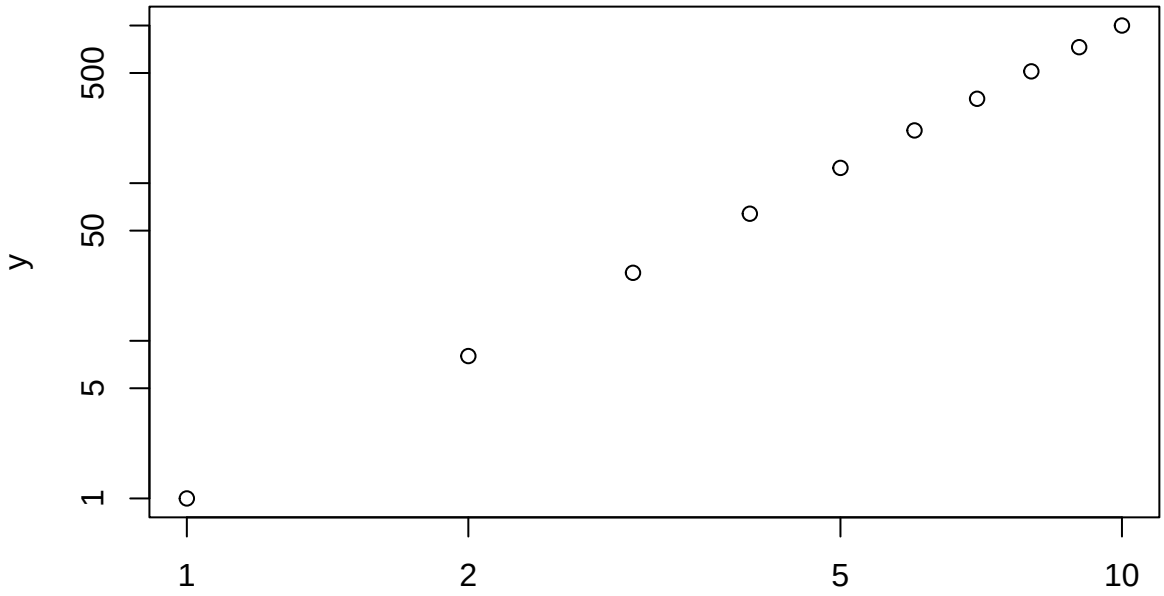
```
x <- c(1:10)
plot(x, power3(x,2), log = "xy", xlab = "log of x", ylab = "log of x^2", main = "log(x^2)~log(x)")
```



Answer:

(f)

```
PlotPower <- function(x,a){
  y <- power3(x,a)
  plot(x,y,log = "xy")
}
PlotPower(1:10, 3)
```



Answer:

4.16

```
summary(Boston)
```

Answer:

```
##       crim                zn             indus            chas        
##  Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   Min.   :0.00000  
##  1st Qu.: 0.08205   1st Qu.:  0.00   1st Qu.: 5.19   1st Qu.:0.00000  
##  Median : 0.25651   Median :  0.00   Median : 9.69   Median :0.00000  
##  Mean   : 3.61352   Mean   : 11.36   Mean   :11.14   Mean   :0.06917  
##  3rd Qu.: 3.67708   3rd Qu.: 12.50   3rd Qu.:18.10   3rd Qu.:0.00000  
##  Max.   :88.97620   Max.   :100.00   Max.   :27.74   Max.   :1.00000  
##       nox               rm              age              dis        
##  Min.   :0.3850   Min.   :3.561   Min.   :  2.90   Min.   : 1.130  
##  1st Qu.:0.4490   1st Qu.:5.886   1st Qu.: 45.02   1st Qu.: 2.100  
##  Median :0.5380   Median :6.208   Median : 77.50   Median : 3.207  
##  Mean   :0.5547   Mean   :6.285   Mean   : 68.57   Mean   : 3.795  
##  3rd Qu.:0.6240   3rd Qu.:6.623   3rd Qu.: 94.08   3rd Qu.: 5.188  
##  Max.   :0.8710   Max.   :8.780   Max.   :100.00   Max.   :12.127  
##       rad              tax           ptratio          black       
##  Min.   : 1.000   Min.   :187.0   Min.   :12.60   Min.   :  0.32  
##  1st Qu.: 4.000   1st Qu.:279.0   1st Qu.:17.40   1st Qu.:375.38  
##  Median : 5.000   Median :330.0   Median :19.05   Median :391.44  
##  Mean   : 9.549   Mean   :408.2   Mean   :18.46   Mean   :356.67  
##  3rd Qu.:24.000   3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:396.23  
##  Max.   :24.000   Max.   :711.0   Max.   :22.00   Max.   :396.90  
##      lstat            medv      
##  Min.   : 1.73   Min.   : 5.00  
##  1st Qu.: 6.95   1st Qu.:17.02  
##  Median :11.36   Median :21.20  
##  Mean   :12.65   Mean   :22.53  
##  3rd Qu.:16.95   3rd Qu.:25.00  
##  Max.   :37.97   Max.   :50.00
```

```
attach(Boston)
crim01 <- rep(0,length(crim))
crim01[crim>median(crim)] <- 1
Boston <- cbind(Boston,crim01)
#test correlation
cor(Boston)
```

```
##                crim          zn       indus         chas         nox
## crim     1.00000000 -0.20046922  0.40658341 -0.055891582  0.42097171
## zn      -0.20046922  1.00000000 -0.53382819 -0.042696719 -0.51660371
## indus    0.40658341 -0.53382819  1.00000000  0.062938027  0.76365145
## chas    -0.05589158 -0.04269672  0.06293803  1.000000000  0.09120281
## nox      0.42097171 -0.51660371  0.76365145  0.091202807  1.00000000
## rm      -0.21924670  0.31199059 -0.39167585  0.091251225 -0.30218819
## age      0.35273425 -0.56953734  0.64477851  0.086517774  0.73147010
## dis     -0.37967009  0.66440822 -0.70802699 -0.099175780 -0.76923011
## rad      0.62550515 -0.31194783  0.59512927 -0.007368241  0.61144056
## tax      0.58276431 -0.31456332  0.72076018 -0.035586518  0.66802320
## ptratio  0.28994558 -0.39167855  0.38324756 -0.121515174  0.18893268
## black   -0.38506394  0.17552032 -0.35697654  0.048788485 -0.38005064
```

```
## lstat    0.45562148 -0.41299457  0.60379972 -0.053929298  0.59087892
## medv    -0.38830461  0.36044534 -0.48372516  0.175260177 -0.42732077
## crim01   0.40939545 -0.43615103  0.60326017  0.070096774  0.72323480
##                 rm         age         dis          rad         tax    ptratio
## crim    -0.21924670  0.35273425 -0.37967009  0.625505145  0.58276431  0.2899456
## zn       0.31199059 -0.56953734  0.66440822 -0.311947826 -0.31456332 -0.3916785
## indus   -0.39167585  0.64477851 -0.70802699  0.595129275  0.72076018  0.3832476
## chas     0.09125123  0.08651777 -0.09917578 -0.007368241 -0.03558652 -0.1215152
## nox     -0.30218819  0.73147010 -0.76923011  0.611440563  0.66802320  0.1889327
## rm       1.00000000 -0.24026493  0.20524621 -0.209846668 -0.29204783 -0.3555015
## age     -0.24026493  1.00000000 -0.74788054  0.456022452  0.50645559  0.2615150
## dis      0.20524621 -0.74788054  1.00000000 -0.494587930 -0.53443158 -0.2324705
## rad     -0.20984667  0.45602245 -0.49458793  1.000000000  0.91022819  0.4647412
## tax     -0.29204783  0.50645559 -0.53443158  0.910228189  1.00000000  0.4608530
## ptratio -0.35550149  0.26151501 -0.23247054  0.464741179  0.46085304  1.0000000
## black    0.12806864 -0.27353398  0.29151167 -0.444412816 -0.44180801 -0.1773833
## lstat   -0.61380827  0.60233853 -0.49699583  0.488676335  0.54399341  0.3740443
## medv     0.69535995 -0.37695457  0.24992873 -0.381626231 -0.46853593 -0.5077867
## crim01  -0.15637178  0.61393992 -0.61634164  0.619786249  0.60874128  0.2535684
##              black      lstat       medv      crim01
## crim    -0.38506394  0.4556215 -0.3883046  0.40939545
## zn       0.17552032 -0.4129946  0.3604453 -0.43615103
## indus   -0.35697654  0.6037997 -0.4837252  0.60326017
## chas     0.04878848 -0.0539293  0.1752602  0.07009677
## nox     -0.38005064  0.5908789 -0.4273208  0.72323480
## rm       0.12806864 -0.6138083  0.6953599 -0.15637178
## age     -0.27353398  0.6023385 -0.3769546  0.61393992
## dis      0.29151167 -0.4969958  0.2499287 -0.61634164
## rad     -0.44441282  0.4886763 -0.3816262  0.61978625
## tax     -0.44180801  0.5439934 -0.4685359  0.60874128
## ptratio -0.17738330  0.3740443 -0.5077867  0.25356836
## black    1.00000000 -0.3660869  0.3334608 -0.35121093
## lstat   -0.36608690  1.0000000 -0.7376627  0.45326273
## medv     0.33346082 -0.7376627  1.0000000 -0.26301673
## crim01  -0.35121093  0.4532627 -0.2630167  1.00000000
```

The nox, dis, rad, age, tax, indus are predictors most associate with probability of crime.

```
train <- 1:400
test <- 401:dim(Boston)[1]
Boston.train <- Boston[train,]
Boston.test <- Boston[test,]
crim01.train <- crim01[train]
crim01.test <- crim01[test]
```

```
##LDA
fit16.lda <- lda(crim01~rad, data=Boston.train)
pred16.lda <- predict(fit16.lda, Boston.test)
mean(pred16.lda$class != crim01.test) #error rate 2.83%
```

```
## [1] 0.02830189
```

```
fit16.lda1 <- lda(crim01~rad+nox, data=Boston.train)
pred16.lda1 <- predict(fit16.lda1, Boston.test)
mean(pred16.lda1$class != crim01.test) #error rate increases
```

```
## [1] 0.09433962
```

```
#add any other predictors, error rate doesn't decrease
```

For LDA, when only use rad, model has lowest error rate, 2.83%.

```
##Logistic
fit16.logit <- glm(crim01~rad, data=Boston.train, family = binomial)
pred16.prob <- predict(fit16.logit, Boston.test)
pred16.logit <- rep(0, length(pred16.prob))
pred16.logit[pred16.prob>0.5] <- 1
mean(pred16.logit != crim01.test)
```

```
## [1] 0.02830189
```

Similarly with LDA, when only use rad, model has lowest error rate, 2.83%.

```
##Naive Bayes
fit16.nb <- naiveBayes(crim01~rad, data=Boston.train)
pred16.nb <- predict(fit16.nb, Boston.test)
mean(pred16.nb != crim01.test)
```

```
## [1] 0.02830189
```

```
fit16.nb1 <- naiveBayes(crim01~rad+indus, data=Boston.train)
pred16.nb1 <- predict(fit16.nb1, Boston.test)
mean(pred16.nb1 != crim01.test)
```

```
## [1] 0.0754717
```

For Naive Bayes, when only use rad, model has lowest error rate, 2.83%.

Add any other predictors, error rate doesn't decrease.

```
##KNN
train16.x <- as.matrix(rad[train])
test16.x <- as.matrix(rad[test])
set.seed(1)
pred16.knn <- knn(train16.x, test16.x, crim01.train, k=200)
mean(pred16.knn != crim01.test)
```

```
## [1] 0.02830189
```

```
train16.x1 <- cbind(rad,nox)[train,]
test16.x1 <- cbind(rad,nox)[test,]
set.seed(1)
pred16.knn1 <- knn(train16.x1, test16.x1, crim01.train, k=100)
mean(pred16.knn1 != crim01.test)
```

```
## [1] 0.02830189
```

For any K<200, the error rate of KNN with only one predictor "rad" remain constant 2.83%.