MA685 HW3

Jiayuan Shi

Feb 20, 2016

Exercise 1 (Conceptional: Training and Test Error)

Exercise 8 (p. 170): Compare logistic regression and KNN based on error rates.

For KNN with K=1, we have a training error rate of 0%, because for any training observation, its nearest neighbor will be the response itself, and we do not make any error on the training data. However, we have an average error rate of 18%, so KNN has a test error rate of 36%, which is greater than the test error rate for logistic regression of 30%. Based on these results, we should prefer logistic regression because of its lower test error rate.

Exercise 2 (Conceptional: Odds)

Exercise 9 (p. 170): Interpretation using odds.

(a)

$$\begin{aligned} odds = \frac{p(X)}{1-p(X)} &= 0.37 \\ p(X) &= 0.37(1-p(X)) \\ 1.37p(X) &= 0.37 \\ p(X) &= \frac{0.37}{1.37} = 27\% \end{aligned}$$

On average, 27% of people with an odds of 0.37 of defaulting on their credit card payment will in fact default.

(b)

$$odds = \frac{p(X)}{1-p(X)} = \frac{0.16}{1-0.16} = 19\%$$

The odds that she will default is 19%.

Exercise 3 (Applied: Comparison of Classification Methods I)

Exercise 11 (p. 171): Perform a comparison of classification methods using the Auto data set.

(a)

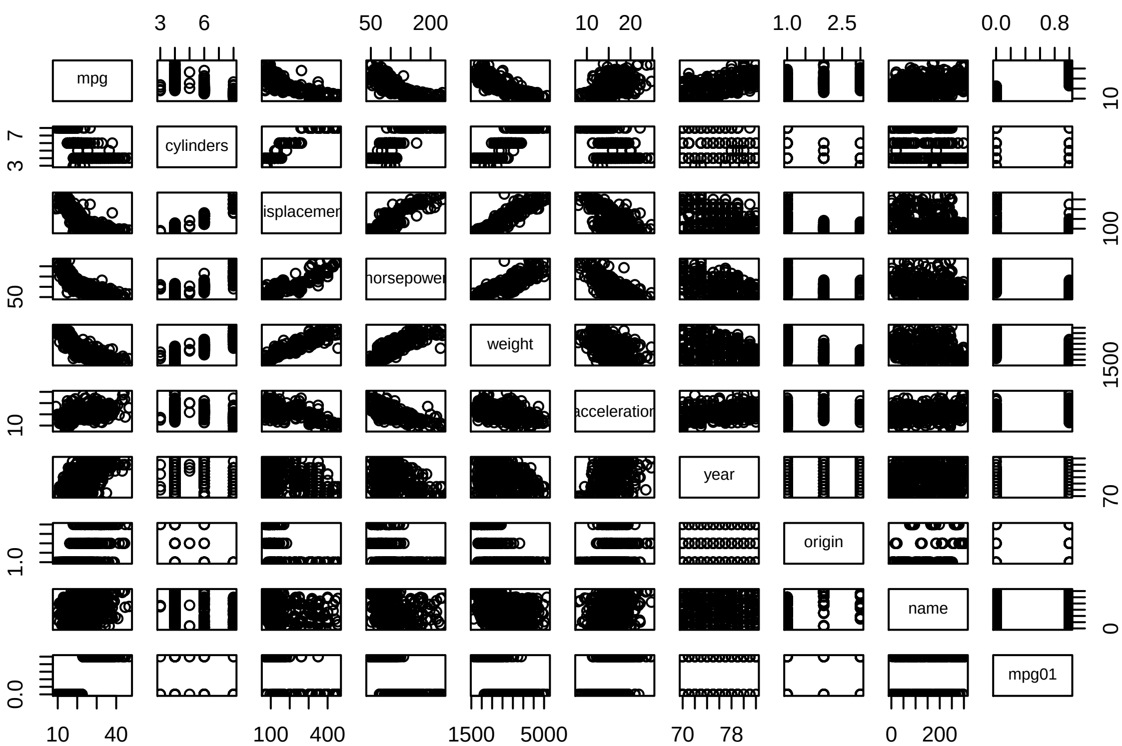
```
library(ISLR)
attach(Auto)
mpg01 <- rep(0, length(mpg))
mpg01[mpg > median(mpg)] <- 1
Auto <- data.frame(Auto, mpg01)
```

(b)

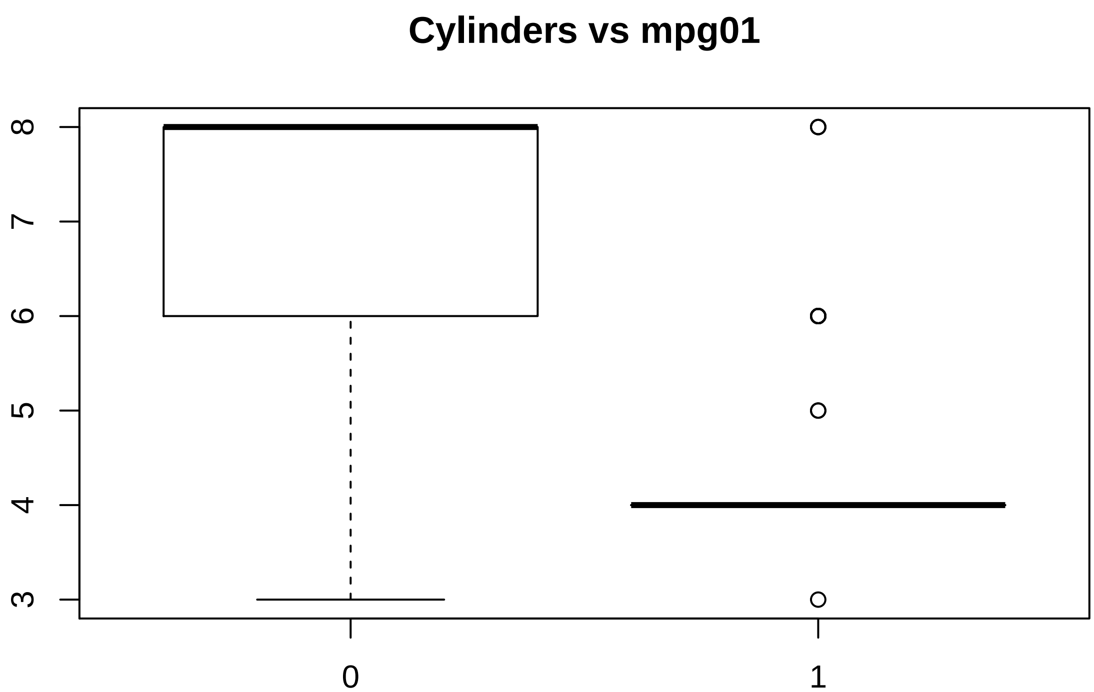
```r
cor(Auto[,-9])
```

```
##                     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
```

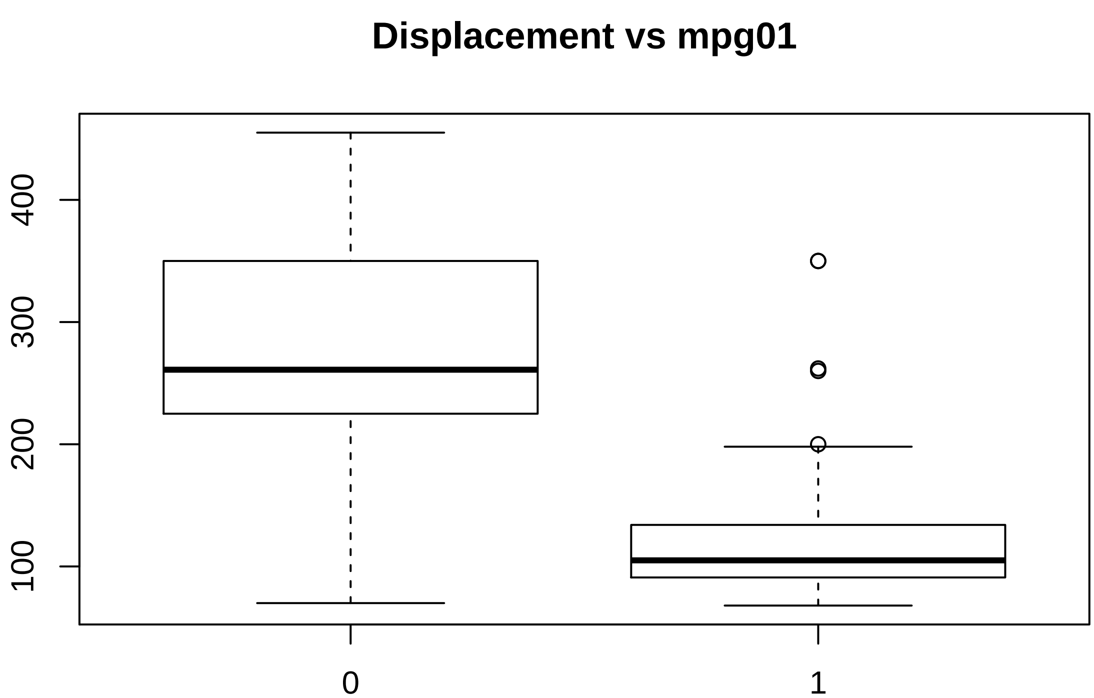
```r
pairs(Auto)
```



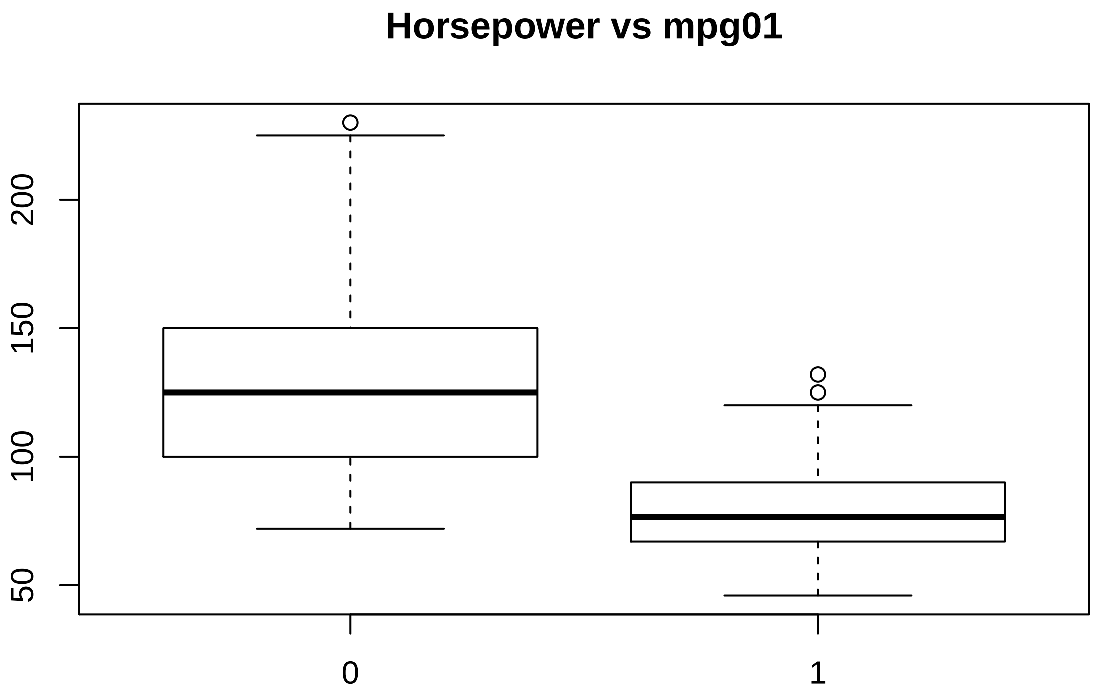
```r
boxplot(cylinders ~ mpg01, data = Auto, main = "Cylinders vs mpg01")
```



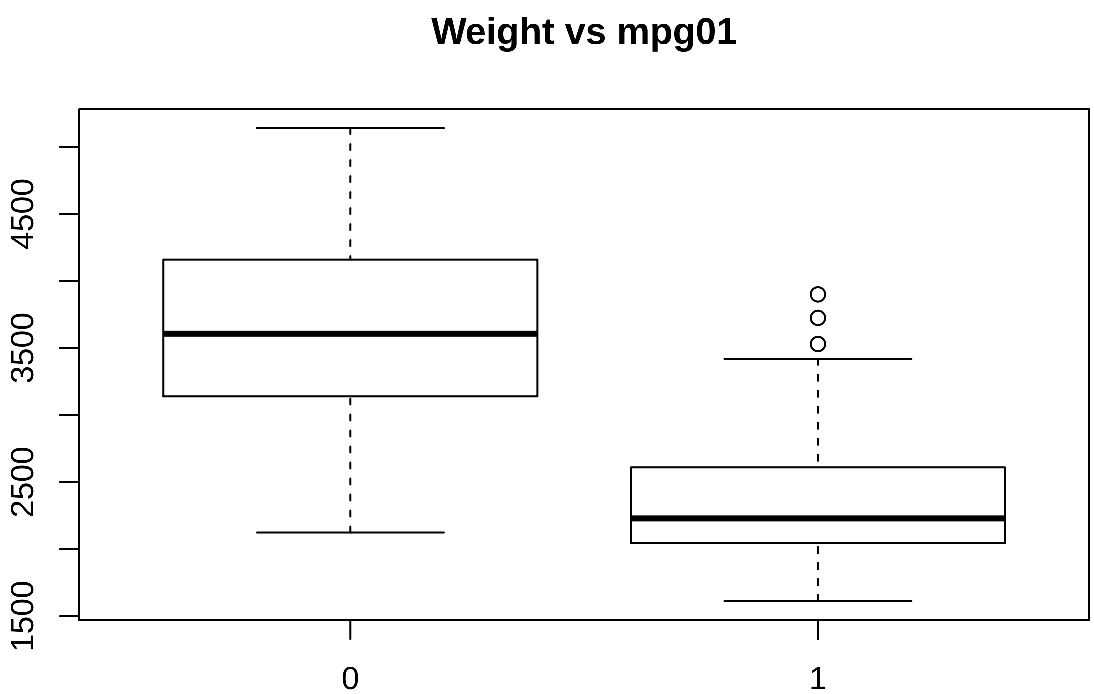

```r
boxplot(displacement ~ mpg01, data = Auto, main = "Displacement vs mpg01")
```



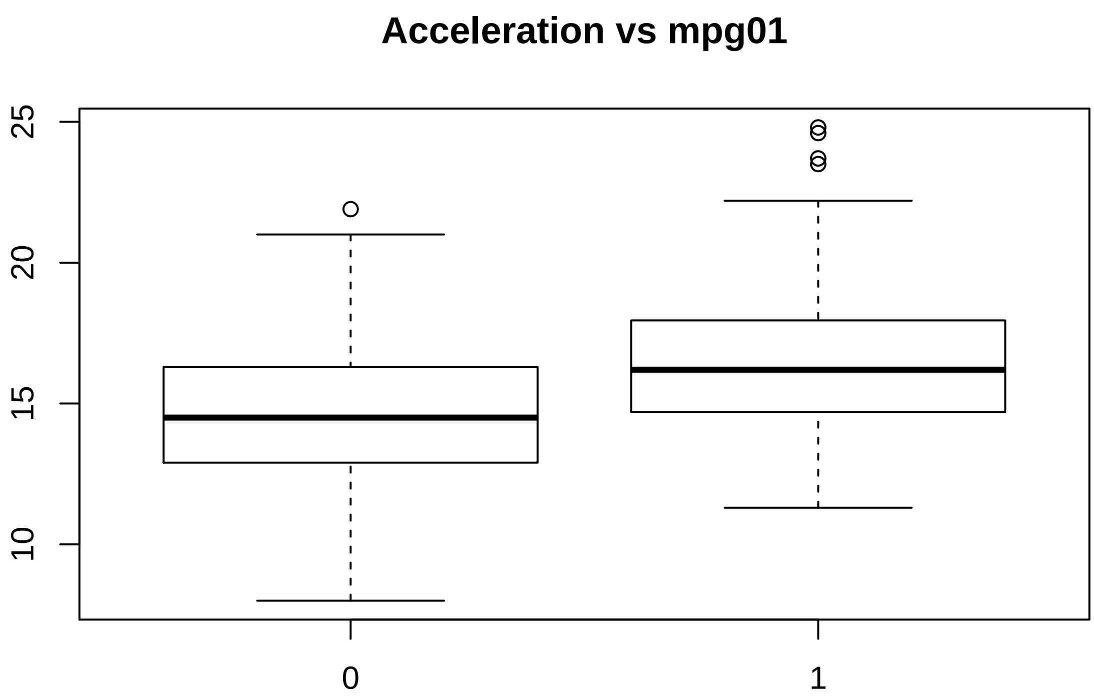

```r
boxplot(horsepower ~ mpg01, data = Auto, main = "Horsepower vs mpg01")
```



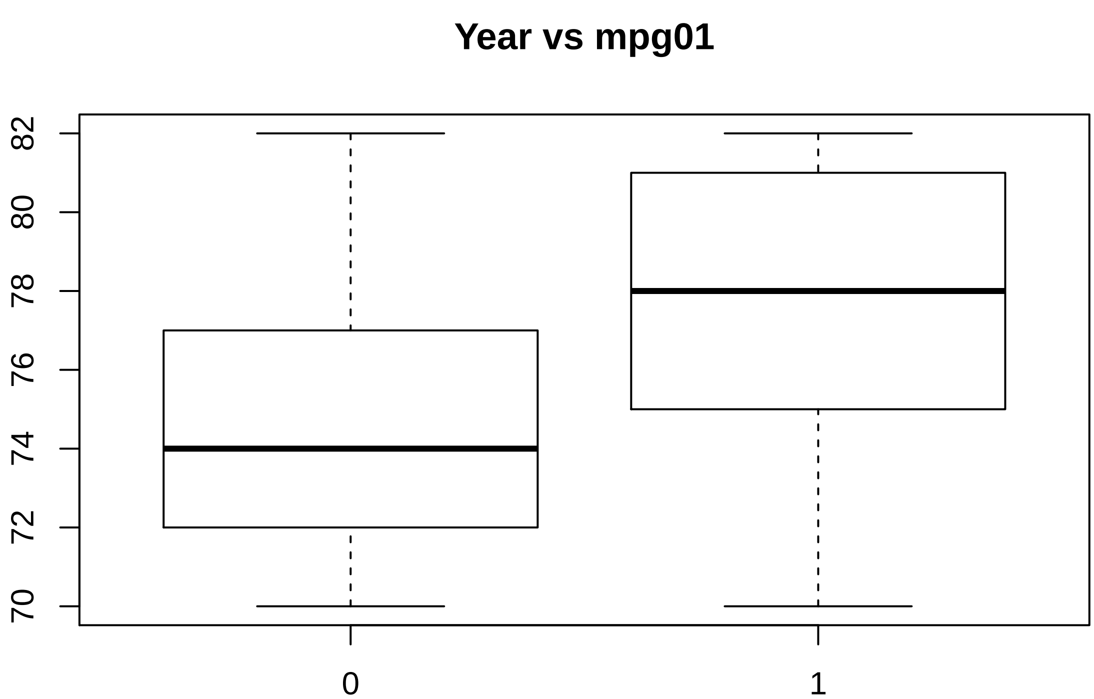

```r
boxplot(weight ~ mpg01, data = Auto, main = "Weight vs mpg01")
```




```r
boxplot(acceleration ~ mpg01, data = Auto, main = "Acceleration vs mpg01")
```




```
boxplot(year ~ mpg01, data = Auto, main = "Year vs mpg01")
```



There exists some association between mpg01 and cylinders, displacement, horsepower, weight.

(c)

```
train <- (year %% 2 == 0)
Auto.train <- Auto[train, ]
```

```r
Auto.test <- Auto[!train, ]
mpg01.test <- mpg01[!train]
```

(d)

```r
# LDA
library(MASS)
fit.lda <- lda(mpg01 ~ cylinders+weight+displacement+horsepower,
              data=Auto, subset=train)
fit.lda
```

```
## Call:
## lda(mpg01 ~ cylinders + weight + displacement + horsepower, data = Auto,
##     subset = train)
##
## Prior probabilities of groups:
##         0         1
## 0.4571429 0.5428571
##
## Group means:
##   cylinders   weight displacement horsepower
## 0  6.812500 3604.823     271.7396  133.14583
## 1  4.070175 2314.763     111.6623   77.92105
##
## Coefficients of linear discriminants:
##                        LD1
## cylinders    -0.6741402638
## weight       -0.0011465750
## displacement  0.0004481325
## horsepower    0.0059035377
```

```r
pred.lda <- predict(fit.lda, Auto.test)
mean(pred.lda$class != mpg01.test)
```

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

The test error of the model obtained is 12.64%

(e)

```r
# QDA
fit.qda <- qda(mpg01~cylinders+weight+displacement+horsepower,
              data=Auto, subset=train)
fit.qda
```

```
## Call:
## qda(mpg01 ~ cylinders + weight + displacement + horsepower, data = Auto,
##     subset = train)
```

```
##
## Prior probabilities of groups:
##         0         1
## 0.4571429 0.5428571
##
## Group means:
##   cylinders   weight displacement horsepower
## 0  6.812500 3604.823     271.7396  133.14583
## 1  4.070175 2314.763     111.6623   77.92105
```

```
pred.qda <- predict(fit.qda, Auto.test)
mean(pred.qda$class != mpg01.test)
```

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

The test error of the model obtained is 13.19%

(f)

```
fit.glm <- glm(mpg01~cylinders+weight+displacement+horsepower,
               data=Auto, family=binomial, subset=train)
summary(fit.glm)
```

```
##
## Call:
## glm(formula = mpg01 ~ cylinders + weight + displacement + horsepower,
##     family = binomial, data = Auto, subset = train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.48027  -0.03413   0.10583   0.29634   2.57584
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  17.658730   3.409012   5.180 2.22e-07 ***
## cylinders    -1.028032   0.653607  -1.573   0.1158
## weight       -0.002922   0.001137  -2.569   0.0102 *
## displacement  0.002462   0.015030   0.164   0.8699
## horsepower   -0.050611   0.025209  -2.008   0.0447 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 289.58  on 209  degrees of freedom
## Residual deviance:  83.24  on 205  degrees of freedom
## AIC: 93.24
##
## Number of Fisher Scoring iterations: 7
```

```
probs <- predict(fit.glm, Auto.test, type="response")
pred.glm <- rep(0, length(probs))
pred.glm[probs>0.5] <- 1
mean(pred.glm != mpg01.test)
```

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

The test error of the model obtained is 12.09%

(g)

```
library(class)
train.X <- cbind(cylinders, weight, displacement, horsepower)[train, ]
test.X <- cbind(cylinders, weight, displacement, horsepower)[!train, ]
train.mpg01 <- mpg01[train]
set.seed(1)
pred.knn <- knn(train.X, test.X, train.mpg01, k=1)
mean(pred.knn != mpg01.test)
```

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

```
pred.knn = knn(train.X, test.X, train.mpg01, k=10)
mean(pred.knn != mpg01.test)
```

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

```
pred.knn = knn(train.X, test.X, train.mpg01, k=100)
mean(pred.knn != mpg01.test)
```

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

K=1, I obtain 15.38% test error rate. K=10, I obtain 16.48% test error rate. K=100, I obtain 14.29% test error rate. A K vlaue of 100 seems to perform the best.

Exercise 4 (Applied: Comparison of Classification Methods II)

Exercise 13 (p. 173): Perform a comparison of classification methods using the Boston data set.

```
library(MASS)
attach(Boston)
crim01 <- rep(0, length(crim))
crim01[crim > median(crim)] <- 1
Boston <- data.frame(Boston, crim01)

train <- 1:(length(crim)/2)
test <- (length(crim)/2+1):length(crim)
Boston.train <- Boston[train,]
```

```r
Boston.test <- Boston[test,]
crim01.test <- crim01[test]

# logistic regression
fit.glm <- glm(crim01~.-crim01-crim, data=Boston, family=binomial, subset=train)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
probs <- predict(fit.glm, Boston.test, type = "response")
pred.glm <- rep(0, length(probs))
pred.glm[probs>0.5] <- 1
table(pred.glm, crim01.test)
```

```
##         crim01.test
## pred.glm   0   1
##        0  68  24
##        1  22 139
```

```r
mean(pred.glm != crim01.test)
```

```
## [1] 0.1818182
```

```r
fit.glm <- glm(crim01~.-crim01-crim-chas-tax, data=Boston, family=binomial, subset=train)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
probs <- predict(fit.glm, Boston.test, type = "response")
pred.glm <- rep(0, length(probs))
pred.glm[probs>0.5] <- 1
mean(pred.glm != crim01.test)
```

```
## [1] 0.1857708
```

For the logistic regression, with various subsets of the predictors, I get test error rate of 18.18% and 18.58%.

```r
# LDA
fit.lda <- lda(crim01~.-crim01-crim, data=Boston, family=binomial, subset=train)
pred.lda <- predict(fit.lda, Boston.test)
mean(pred.lda$class != crim01.test)
```

```
## [1] 0.1343874
```

```r
fit.lda <- lda(crim01~.-crim01-crim-chas-tax, data=Boston, family=binomial, subset=train)
pred.lda <- predict(fit.lda, Boston.test)
mean(pred.lda$class != crim01.test)
```

```
## [1] 0.1225296
```

For the LDA, with various subsets of the predictors, I get test error rate of 13.44% and 12.25%.

```
train.X <- cbind(zn, indus, chas, nox, rm, age, dis, rad, tax, ptratio, black, lstat, medv)[train,]
test.X <- cbind(zn, indus, chas, nox, rm, age, dis, rad, tax, ptratio, black, lstat, medv)[test,]
train.crim01 <- crim01[train]
set.seed(1)
pred.knn <- knn(train.X, test.X, train.crim01, k=1)
mean(pred.knn != crim01.test)
```

```
## [1] 0.458498
```

```
pred.knn <- knn(train.X, test.X, train.crim01, k=10)
mean(pred.knn != crim01.test)
```

```
## [1] 0.1185771
```

```
pred.knn <- knn(train.X, test.X, train.crim01, k=100)
mean(pred.knn != crim01.test)
```

```
## [1] 0.4901186
```

K=1, I obtain 45.85% test error rate. K=10, I obtain 11.86% test error rate. K=100, I obtain 49.01% test error rate. A K vlaue of 10 seems to perform the best.

```
# KNN(k=10) with subset of variables
train.X <- cbind(zn, nox, rm, dis, rad, ptratio, black, medv)[train,]
test.X <- cbind(zn, nox, rm, dis, rad, ptratio, black, medv)[test,]
train.crim01 <- crim01[train]
set.seed(1)
pred.knn <- knn(train.X, test.X, train.crim01, k=10)
mean(pred.knn != crim01.test)
```

```
## [1] 0.2766798
```

K=10, with subsets of the predictors, I obtain 27.67% test error rate.