

STAT 8330 FALL 2015 ASSIGNMENT 1

Peng Shao

September 6, 2015

► Exercises 2.5. Solution.

(1).

- advantage: can fit many different functional forms; low bias; usually predict more accurately
- disadvantage: overfitting problem; usually hard to interpret; high variance

- (2). If our goal is to predict more accurately, it will usually be best to choose a more flexible approach.
(3). If our goal is to make some inferences, we prefer choosing a less flexible approach because the relation between response and predictor is more explicit.

► Exercises 2.6. Solution.

(1). The essential difference between parametric and non-parametric approach is that, the parametric make an assumption of the form of f , which can reduce problem of estimating f down to one of estimating a set of parameter, but non-parametric do not make explicit assumptions about the functional form of f .

(2).

- advantage: it is easier to estimate parameter; the relation between response and predictor is more explicit;
- disadvantage: the model we choose will usually not match the true unknown form of f ; sometimes need more assumption.

► Exercises 2.10. Solution.

(a).

```
library(MASS)
attach(Boston)
?Boston
```

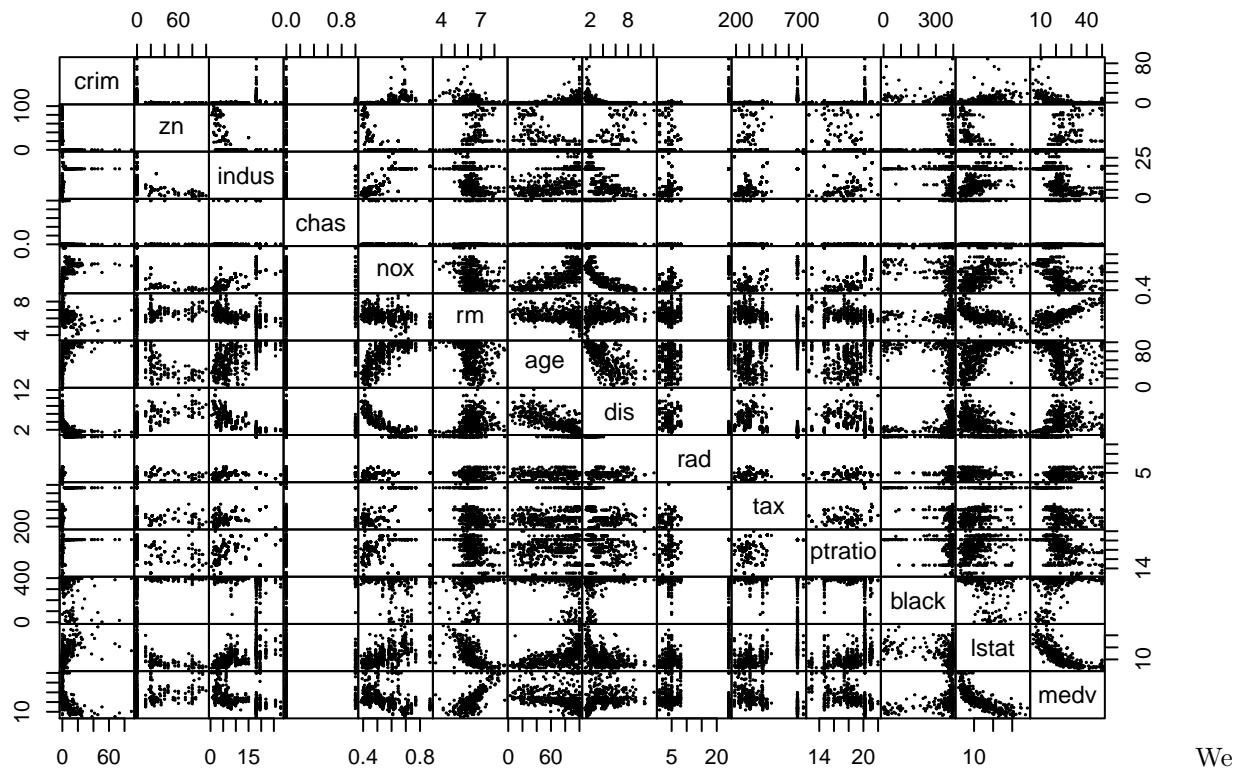
The Boston data frame has 506 rows and 14 columns. Each row represents an observation of a town. The

meaning of columns are

crim : per capita crime rate by town
zn : proportion of residential land zoned for lots over 25,000 sq.ft
indus : proportion of non-retail business acres per town
chas : Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
nox : nitrogen oxides concentration (parts per 10 million)
rm : average number of rooms per dwelling
age : proportion of owner-occupied units built prior to 1940
dis : weighted mean of distances to five Boston employment centres
rad : index of accessibility to radial highways
tax : full-value property-tax rate per \$10,000
ptratio : pupil-teacher ratio by town
black : $1000(Bk - 0.63)^2$ where Bk is the proportion of blacks by town
lstat : lower status of the population (percent)
medv : median value of owner-occupied homes in \$1000s

(b).

```
pairs(Boston, pch = 20, gap = 0, cex = 0.2)
```



We can see that nox, age and dis seems to have strong relationship with each other, and rm, lstat and medv also have strong relationship in the plots. The other pattern seem not to be that apparent.

(c).

```
cor(Boston[, 1], Boston[, -1])
```

##	zn	indus	chas	nox	rm	age
----	----	-------	------	-----	----	-----

```
## [1,] -0.2004692 0.4065834 -0.05589158 0.4209717 -0.2192467 0.3527343
##          dis      rad      tax     ptratio     black     lstat
## [1,] -0.3796701 0.6255051 0.5827643 0.2899456 -0.3850639 0.4556215
##          medv
## [1,] -0.3883046
```

We can see that the rad and tax are relatively highly correlated with crime rate, with 0.63 and 0.58 correlation coefficient respectively. Both of these two correlations are positive.

(d). Yes. No. Almost No.

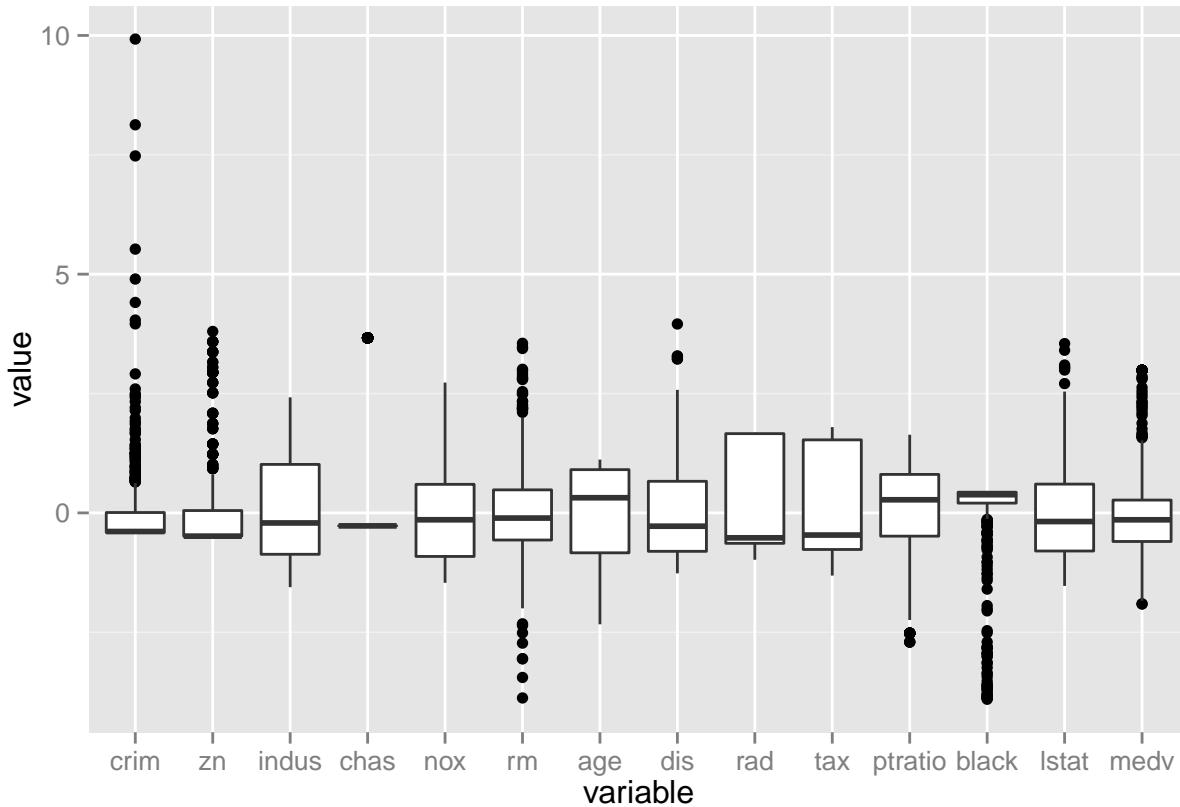
From the summaries and the boxplots, we can know that crim, zn and medv are right skewed; black is left skewed; rm is fat-tailed.

```
summary(Boston)
```

```
##      crim            zn            indus           chas
##  Min.   : 0.00632   Min.   : 0.00   Min.   : 0.46   Min.   :0.00000
##  1st Qu.: 0.08204   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
```

```
library(ggplot2)
library(reshape)
B <- melt(as.data.frame(scale(Boston)))
```

```
## Using as id variables
ggplot(B,aes(x = variable,y = value)) + geom_boxplot()
```



(e)

```
sum(chas == 1)
```

```
## [1] 35
```

(f) From part (d).., the median of pupil-teacher ratio is 19.50.

(g)

```
ind <- which(medv == min(medv))
ind
```

```
## [1] 399 406
```

```
Boston[ind, ]
```

```
##      crim  zn  indus  chas    nox     rm   age     dis   rad   tax  ptratio   black
## 399 38.3518  0 18.1    0 0.693 5.453 100 1.4896  24 666    20.2 396.90
## 406 67.9208  0 18.1    0 0.693 5.683 100 1.4254  24 666    20.2 384.97
##      lstat medv
## 399 30.59    5
## 406 22.98    5
```

The suburbs with lowest median value of owneroccupied homes also have relatively high crime, large population of lower status and proportion of non-retail business acres, which indicates that these suburbs are kind of less developed areas.

(i) The number of the suburbs average more than seven rooms per dwelling is:

```
sum(rm > 7)
```

```
## [1] 64
```

The number of the suburbs average more than eight rooms per dwelling is:

```
sum(rm > 8)
```

```
## [1] 13
```

Because the maximum of rm is 8.780, then the the suburbs average more than eight rooms per dwelling can be marked as outlier for this dataset.

► Exercises 3.5. Solution.

It is very easy to get the result by using hat matrix. Consider the model

$$\mathbf{Y} = \mathbf{X}\beta$$

where $\mathbf{Y} = [y_1, \dots, y_n]^T$, $\mathbf{X} = [x_1, \dots, x_n]^T$. Then

$$\hat{\beta} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y} = \left(\sum_{i=1}^n x_i y_i \right) / \left(\sum_{j=1}^n x_j^2 \right)$$

so

$$\hat{Y} = \mathbf{X} \hat{\beta} \left(\sum_{i=1}^n x_i y_i \right) / \left(\sum_{j=1}^n x_j^2 \right)$$

and

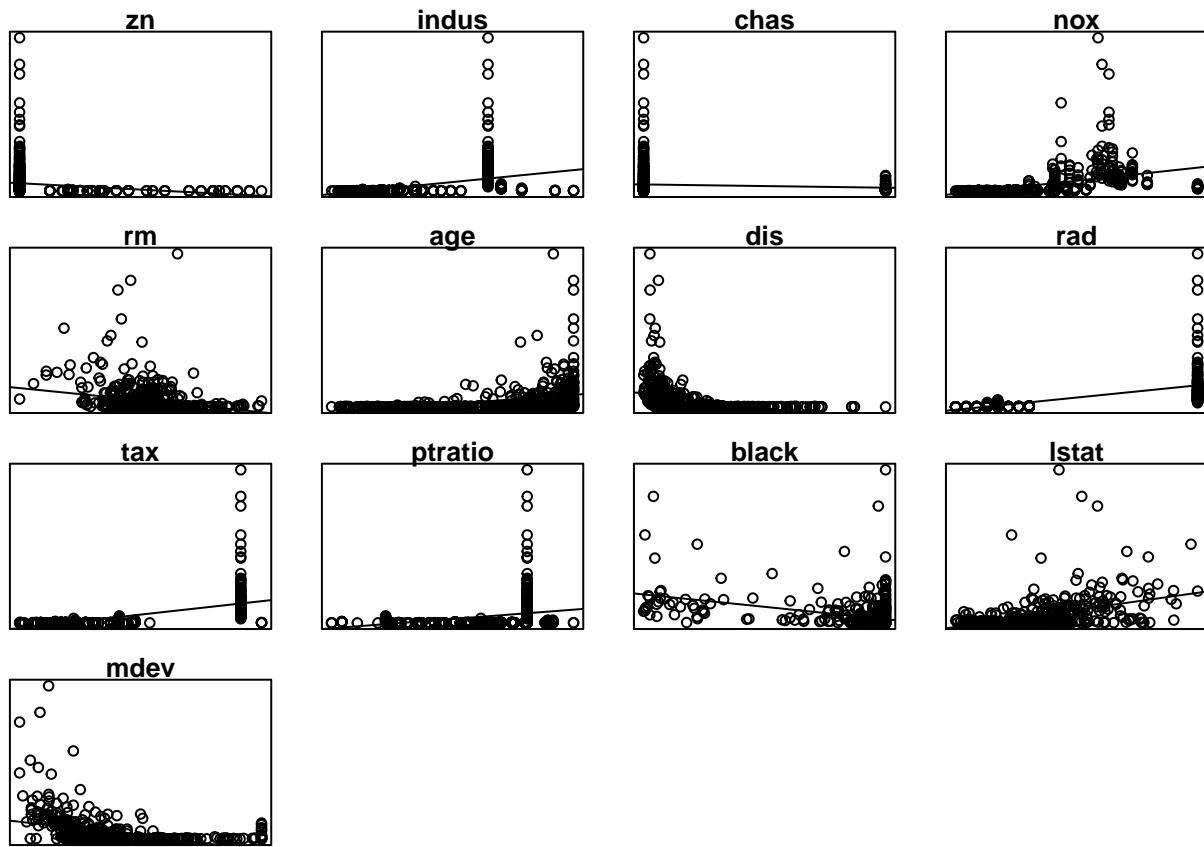
$$\hat{y}_i = x_i \left(\sum_{i'=1}^n x_{i'} \right) / \left(\sum_{j=1}^n x_j^2 \right) y_{i'} = \sum_{i'=1}^n (x_i x_{i'}) / \left(\sum_{j=1}^n x_j^2 \right) y_{i'}$$

Thus,

$$a_{i'} = (x_i x_{i'}) / \left(\sum_{j=1}^n x_j^2 \right)$$

► Exercises 3.15. Solution.

(a). The statistics output is so long, so I omit the full output. From the statistical results, only chas is not significant correlated with crim. From the plots, we can see that indus, nox, rm, dis, rad, tax, black, lstat and medv show relatively apperant relationship with crim.



(b).

```
lm.fit <- lm(formula = crim ~ ., data = Boston)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = crim ~ ., data = Boston)
##
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -9.924 -2.120 -0.353  1.019 75.051 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 17.033228  7.234903  2.354 0.018949 *  
## zn          0.044855  0.018734  2.394 0.017025 *  
## indus      -0.063855  0.083407 -0.766 0.444294    
## chas       -0.749134  1.180147 -0.635 0.525867    
## nox       -10.313535  5.275536 -1.955 0.051152 .  
## rm         0.430131  0.612830  0.702 0.483089    
## age        0.001452  0.017925  0.081 0.935488    
## dis       -0.987176  0.281817 -3.503 0.000502 *** 
## rad        0.588209  0.088049  6.680 6.46e-11 *** 
## tax       -0.003780  0.005156 -0.733 0.463793    
## ptratio    -0.271081  0.186450 -1.454 0.146611    
## black     -0.007538  0.003673 -2.052 0.040702 *  
## lstat      0.126211  0.075725  1.667 0.096208 .
```

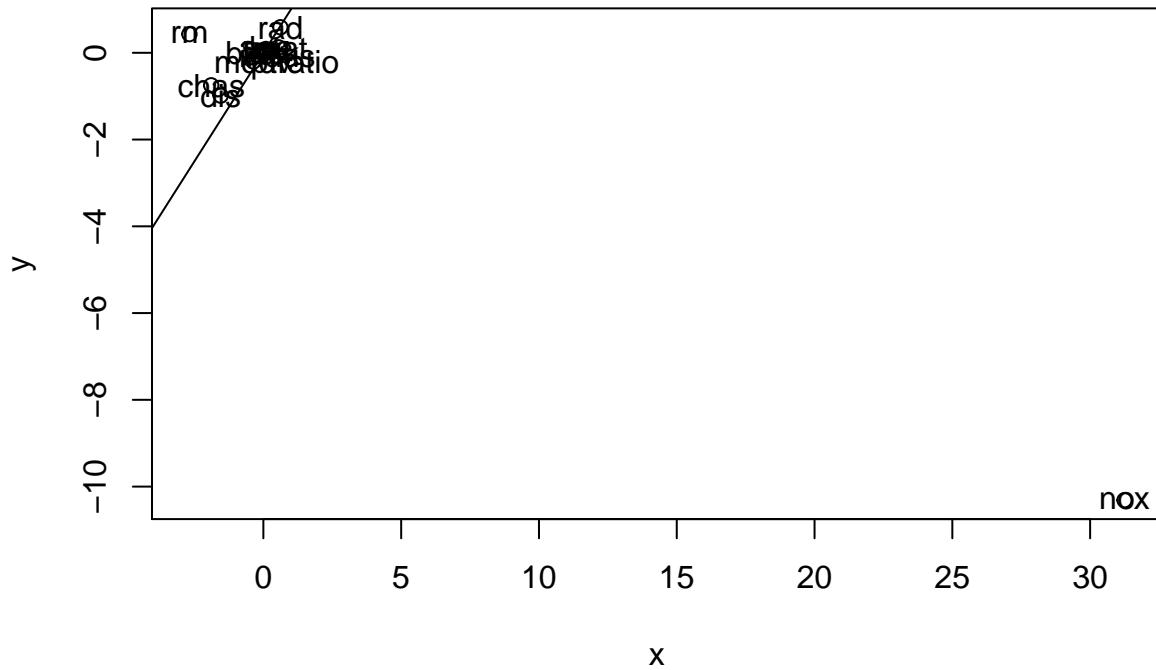
```

## medv      -0.198887  0.060516 -3.287 0.001087 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.439 on 492 degrees of freedom
## Multiple R-squared:  0.454, Adjusted R-squared:  0.4396
## F-statistic: 31.47 on 13 and 492 DF,  p-value: < 2.2e-16
y <- lm.fit$coefficients[-1]

```

Variable `zn`, variable `dis`, variable `rad`, variable `black` and variable `medv` are significant in this model, i.e., they can reject the null hypothesis of zero. Even the F-test shows that the model is useful and significant, the coefficient of determination R^2 is only 0.454, which means less than 50% of data can be explained by this model. We may think that this model is underfitted.

(c). Most coefficients do not change so much, but the coefficients of `nox` are quite different between the single regression model and the multiple regression model. This may be because there is multicollinearity between `nox` and some variables, then the estimate will be greatly affected by the variables which are already included in the model.



(d). Full outputs are omitted because of the length. According to the result of model training, `indus`, `nox`, `age`, `dis`, `ptratio` and `medv` show significant cubic trend. The others

```

lm.fit <- lm(formula = crim ~ zn + I(zn^2) + I(zn^3), data = Boston)
summary(lm.fit)

lm.fit <- lm(formula = crim ~ indus + I(indus^2) + I(indus^3), data = Boston)
summary(lm.fit)

lm.fit <- lm(formula = crim ~ chas + I(chas^2) + I(chas^3), data = Boston)
summary(lm.fit)

lm.fit <- lm(formula = crim ~ nox + I(nox^2) + I(nox^3), data = Boston)
summary(lm.fit)

lm.fit <- lm(formula = crim ~ rm + I(rm^2) + I(rm^3), data = Boston)
summary(lm.fit)

lm.fit <- lm(formula = crim ~ age + I(age^2) + I(age^3), data = Boston)
summary(lm.fit)

```

```

lm.fit <- lm(formula = crim ~ dis + I(dis^2) + I(dis^3), data = Boston)
summary(lm.fit)
lm.fit <- lm(formula = crim ~ rad + I(rad^2) + I(rad^3), data = Boston)
summary(lm.fit)
lm.fit <- lm(formula = crim ~ tax + I(tax^2) + I(tax^3), data = Boston)
summary(lm.fit)
lm.fit <- lm(formula = crim ~ ptratio + I(ptratio^2) + I(ptratio^3), data = Boston)
summary(lm.fit)
lm.fit <- lm(formula = crim ~ black + I(black^2) + I(black^3), data = Boston)
summary(lm.fit)
lm.fit <- lm(formula = crim ~ lstat + I(lstat^2) + I(lstat^3), data = Boston)
summary(lm.fit)
lm.fit <- lm(formula = crim ~ medv + I(medv^2) + I(medv^3), data = Boston)
summary(lm.fit)

```

► Exercises 4.3. Solution.

We know that we classify X into k th class based on Bayes' classifier if

$$p_k(x) = \frac{f_k(x)\pi_k}{\sum_{l=1}^K f_l(x)\pi_l}$$

is largest among all $p_l(x)$, $l = 1, 2, \dots, K$. For 1 dimension, the density of x from k th class is

$$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k} e^{-\frac{(x-\mu_k)^2}{2\sigma_k^2}}$$

In comparing two classes k and l , it is sufficient to look at the log-ratio, and we see that

$$\begin{aligned} \log\left(\frac{p_k(x)}{p_l(x)}\right) &= \log\left(\frac{\pi_k}{\pi_l}\right) + \log\left(\frac{f_k(x)}{f_l(x)}\right) \\ &= \log\left(\frac{\pi_k}{\pi_l}\right) + \log\left(\frac{\sigma_l}{\sigma_k}\right) - \frac{(x-\mu_k)^2}{2\sigma_k^2} + \frac{(x-\mu_l)^2}{2\sigma_l^2} \\ &= \left(-\frac{(x-\mu_k)^2}{2\sigma_k^2} - \log\sigma_k + \log\pi_k\right) - \left(-\frac{(x-\mu_l)^2}{2\sigma_l^2} - \log\sigma_l + \log\pi_l\right) \\ &= \delta_k(x) - \delta_l(x) \end{aligned}$$

Then the Bayes' classifier can be defined as

$$C(x) = \arg \max_k \delta_k(x)$$

where $\delta_k(x) = -\frac{(x-\mu_k)^2}{2\sigma_k^2} - \log\sigma_k + \log\pi_k$.

It is obvious that the decision boundary between each pair of classes k and l is described by a quadratic equation $\{x : \delta_k(x) = \delta_l(x)\}$.

► Exercises 4.10. Solution.

(a). From the output, we can see that (1) the variable Volume is increased as the Year increased, and the increase rates become larger and larger; (2) the variable Today is highly, but not complete, correlated with the indicator variable Direction, so we may guess that Direction is transformed from Today. Except this two pairs, no other pairs show any obvious patterns.

```

## 
## Attaching package: 'class'
## 
```

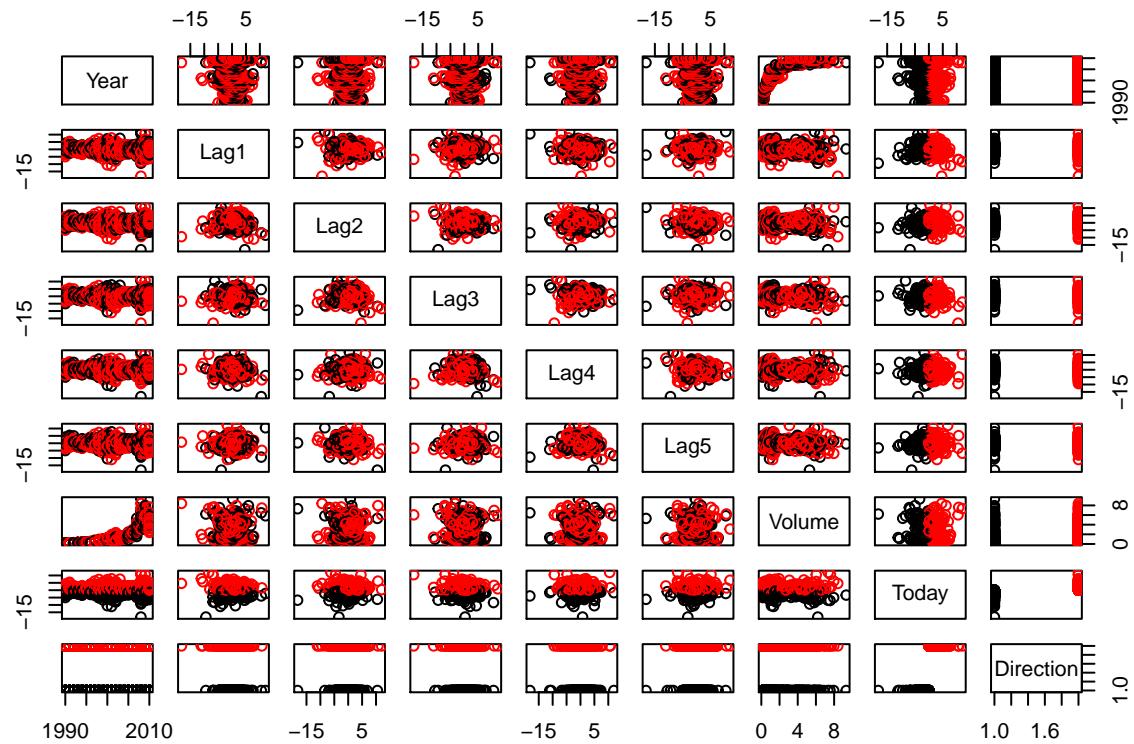
```

## The following object is masked from 'package:reshape':
##
##      condense
##
## Type 'citation("pROC")' for a citation.
##
## Attaching package: 'pROC'
##
## The following objects are masked from 'package:stats':
##
##      cov, smooth, var
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

pairs(Weekly, col = Direction)

```



(b). Fitting the model as below, the summary result shows that only intercept and coefficient of Lag2 is significant.

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

```
##
## 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
```

(c). Using threshold = 0.5, we can get the confusion table as below. At a first glance, the prediction accuracy is 56.11% and the error rate is 43.89%, which is not so good but acceptable, since the prediction of trend of stock index is so difficult. The true positive rate is so good as 92.07%, but the false positive is also too high (almost 90%), which is catastrophic. So we should review the accuracy and error rate. Suppose we have a trivial classifier which always predict "UP". Then the error rate of this classifier is $484/1089 \times 100\% = 44.44\%$, which is just slightly worse than the logistic regression! So, the logistic regression using all variables as predictors is almost useless for this case.

```
glm.probs <- predict(logit.fit, type="response")
glm.pred <- rep("Down", nrow(Weekly))
glm.pred[glm.probs > 0.50] <- "Up"
ct <- table(glm.pred, Direction)
ct
```

```
##          Direction
## glm.pred Down  Up
##        Down   54  48
##        Up    430 557
```

```
ConfusionTable(ct)
```

```
## $Accuracy
## [1] 0.5610652
##
## $`True Positive Rate`
## [1] 0.9206612
##
## $`False Positive Rate`
## [1] 0.8884298
##
## $`Total Error Rate`
## [1] 0.4389348
```

(d). The overall fraction of correct predictions is the accuracy of the classifier, which is 62.5%.

```
train <- Year <= 2008
Weekly.test <- Weekly[!train, ]
logit.fit <- glm(Direction ~ Lag2, family = binomial, data = Weekly, subset = train)
glm.probs <- predict(logit.fit, Weekly.test, type="response")
glm.pred <- rep("Down", nrow(Weekly.test))
glm.pred[glm.probs > 0.50] <- "Up"
ct <- table(glm.pred, Weekly.test$Direction)
ct
```

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

```
ConfusionTable(ct)$Accuracy
```

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

(e). The overall fraction of correct predictions is 62.5%.

```
lda.fit <- lda(Direction ~ Lag2, data = Weekly, subset = train)
lda.class <- predict(lda.fit, Weekly.test)$class
ct <- table(lda.class, Weekly.test$Direction)
ct
```

```
##
## lda.class Down Up
##     Down    9  5
##     Up     34 56
ConfusionTable(ct)$Accuracy
```

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

(f). The overall fraction of correct predictions is 58.65%.

```
qda.fit <- qda(Direction ~ Lag2, data = Weekly, subset = train)
qda.class <- predict(qda.fit, Weekly.test)$class
ct <- table(qda.class, Weekly.test$Direction)
ct
```

```
##
## qda.class Down Up
##     Down    0  0
##     Up     43 61
ConfusionTable(ct)$Accuracy
```

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

(g). The overall fraction of correct predictions is 50%.

```
train.X <- matrix(Lag2[train])
test.X <- matrix(Lag2[!train])
train.Direction <- Direction[train]
test.Direction <- Direction[!train]
set.seed(1)
knn.pred <- knn(train.X, test.X, train.Direction, k = 1)
ct <- table(knn.pred, test.Direction)
ct
```

```
##           test.Direction
## knn.pred Down Up
##       Down   21 30
##       Up    22 31
ConfusionTable(ct)$Accuracy
```

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

(h) To compare the these four methods, the simplest way is to plot the ROC curve of each method. Plots are list below, ans R can also provide the AUCs of four methods. We can see that QDA (the third one) has the largest AUC, which is 0.4914. In this respect, we can say that QDA may be the best classifier among these four methods for this dataset.

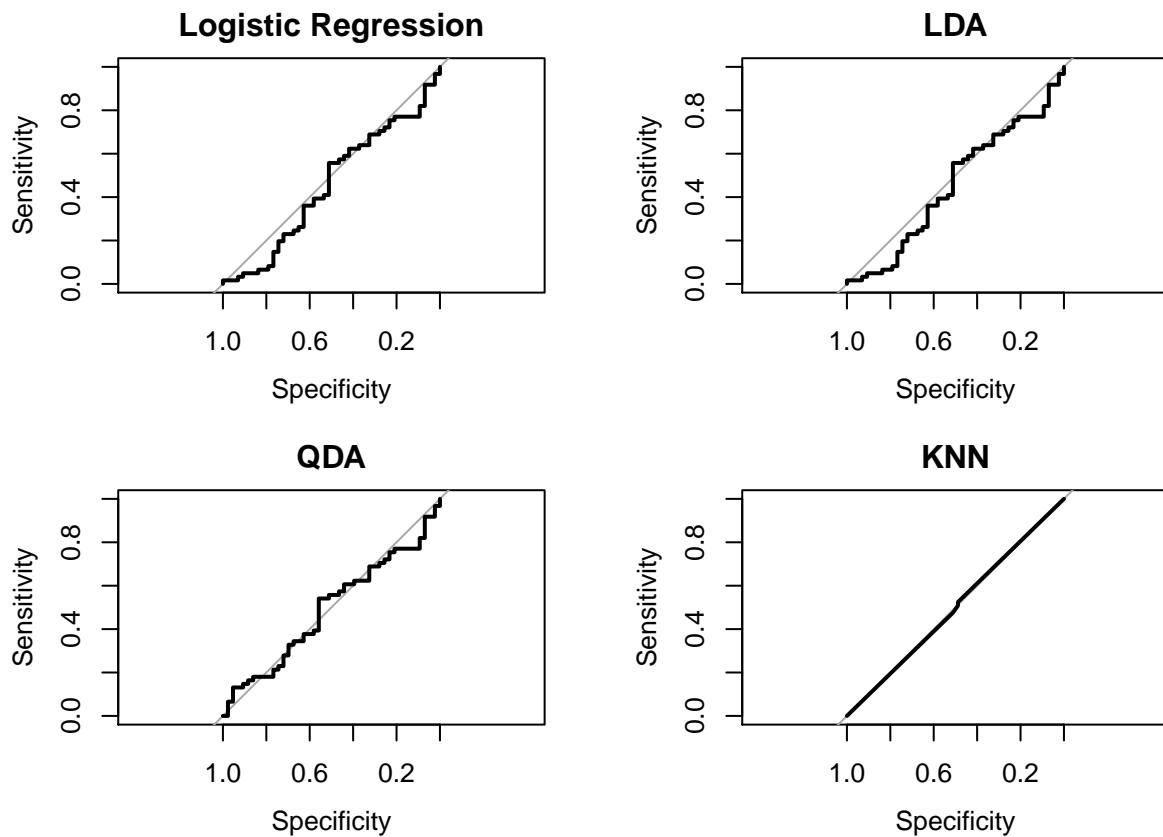
```
##
## Call:
## roc.formula(formula = Direction ~ prob, data = Weekly.test)
##
## Data: prob in 43 controls (Direction Down) > 61 cases (Direction Up).
```

```

## Area under the curve: 0.4537
##
## Call:
## roc.formula(formula = Direction ~ prob, data = Weekly.test)
##
## Data: prob in 43 controls (Direction Down) > 61 cases (Direction Up).
## Area under the curve: 0.4537

##
## Call:
## roc.formula(formula = Direction ~ prob, data = Weekly.test)
##
## Data: prob in 43 controls (Direction Down) > 61 cases (Direction Up).
## Area under the curve: 0.4914

```



```

##
## Call:
## roc.formula(formula = test.Direction ~ prob, data = Weekly.test)
##
## Data: prob in 43 controls (test.Direction Down) < 61 cases (test.Direction Up).
## Area under the curve: 0.4998

```

(i). First, I try different value of K to fit the KNN models.

```

set.seed(1)
knn.pred=knn(train.X,test.X,train.Direction,k=2)
ct <- table(knn.pred, test.Direction)
ct

```

```

##          test.Direction
## knn.pred Down Up
##    Down   19 27
##    Up     24 34
ConfusionTable(ct)

## $Accuracy
## [1] 0.5096154
##
## $`True Positive Rate`
## [1] 0.557377
##
## $`False Posistive Rate`
## [1] 0.5581395
##
## $`Total Error Rate`
## [1] 0.4903846

knn.pred=knn(train.X,test.X,train.Direction,k=4)
ct <- table(knn.pred, test.Direction)
ct

##          test.Direction
## knn.pred Down Up
##    Down   20 20
##    Up     23 41
ConfusionTable(ct)

## $Accuracy
## [1] 0.5865385
##
## $`True Positive Rate`
## [1] 0.6721311
##
## $`False Posistive Rate`
## [1] 0.5348837
##
## $`Total Error Rate`
## [1] 0.4134615

knn.pred=knn(train.X,test.X,train.Direction,k=7)
ct <- table(knn.pred, test.Direction)
ct

##          test.Direction
## knn.pred Down Up
##    Down   15 20
##    Up     28 41
ConfusionTable(ct)

## $Accuracy
## [1] 0.5384615
##
## $`True Positive Rate`
## [1] 0.6721311

```

```
##
## $`False Posistive Rate`
## [1] 0.6511628
##
## $`Total Error Rate`
## [1] 0.4615385
```

We can see that when $K = 4$, the KNN model performances best. But it still has relatively high value of error rates, 41.34%, which is only 2 percents lower than the useless trivial classifier.

For QDA, I try to introduce more variable, beginning with Lag1.

```
Weekly.test <- Weekly[!train, ]
qda.fit <- qda(Direction ~ Lag2 + Lag1, data = Weekly, subset = train)
qda.class <- predict(qda.fit, Weekly.test)$class
ct <- table(qda.class, Weekly.test$Direction)
ct

##
## qda.class Down Up
##     Down    7 10
##     Up      36 51
ConfusionTable(ct)

## $Accuracy
## [1] 0.5576923
##
## $`True Positive Rate`
## [1] 0.8360656
##
## $`False Posistive Rate`
## [1] 0.8372093
##
## $`Total Error Rate`
## [1] 0.4423077
```

Compared to model without Lag1, the full model shows worse performance, so there is no need to introduce more predictors. Similarly, LDA with more predictors also perform worse than that just including Lag2.

For logistic regression, we do not want to introduce more variable, since it does not work on LDA. So we try to introduce the high order term of Lag2, like $Lag2^2$ or $Lag2^3$.

```
logit.fit <- glm(Direction ~ Lag2 + Lag2^2 + Lag2^3, family = binomial, data = Weekly, subset = train)
glm.probs <- predict(logit.fit, Weekly.test, type="response")
glm.pred <- rep("Down", nrow(Weekly.test))
glm.pred[glm.probs > 0.50] <- "Up"
ct <- table(glm.pred, Weekly.test$Direction)
ct

##
## glm.pred Down Up
##     Down    9 5
##     Up      34 56
ConfusionTable(ct)

## $Accuracy
## [1] 0.625
```

```
##
## $`True Positive Rate`
## [1] 0.9180328
##
## $`False Posistive Rate`
## [1] 0.7906977
##
## $`Total Error Rate`
## [1] 0.375
```

The result indicates that there is still no improve in this way.

► Exercises 4.13. Solution.

First, we fit the model with all candidate predictor to see which predictors are significant.

```
## The following objects are masked from Boston (pos = 9):
##
##      age, black, chas, crim, dis, indus, lstat, medv, nox, ptratio,
##      rad, rm, tax, zn
##
## Call:
## glm(formula = crim01 ~ ., family = binomial, data = Boston[, -1])
##
## Deviance Residuals:
##      Min        1Q     Median        3Q       Max
## -2.3946   -0.1585   -0.0004    0.0023    3.4239
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -34.103704   6.530014 -5.223 1.76e-07 ***
## zn          -0.079918   0.033731 -2.369 0.01782 *
## indus       -0.059389   0.043722 -1.358 0.17436
## chas         0.785327   0.728930  1.077 0.28132
## nox          48.523782   7.396497  6.560 5.37e-11 ***
## rm           -0.425596   0.701104 -0.607 0.54383
## age          0.022172   0.012221  1.814 0.06963 .
## dis          0.691400   0.218308  3.167 0.00154 **
## rad          0.656465   0.152452  4.306 1.66e-05 ***
## tax          -0.006412   0.002689 -2.385 0.01709 *
## ptratio      0.368716   0.122136  3.019 0.00254 **
## black        -0.013524   0.006536 -2.069 0.03853 *
## lstat        0.043862   0.048981  0.895 0.37052
## medv         0.167130   0.066940  2.497 0.01254 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 701.46 on 505 degrees of freedom
## Residual deviance: 211.93 on 492 degrees of freedom
## AIC: 239.93
##
## Number of Fisher Scoring iterations: 9
```

According the reuslt, we should treat these seven variables – *zn*, *nox*, *dis*, *rad*, *tax*, *ptratio*, *black*, *medv* – as predictor to build the model.

We first try LDA since it is the most inflexible model among these three method.

```
lda.fit <- lda(crim01 ~ zn + nox + black + dis + rad + tax + ptratio + medv, data=Boston.train)
lda.pred=predict(lda.fit,Boston.test)$class
ct <- table(lda.pred,Boston.test$crim01)
ct

##
## lda.pred 0 1
##      0 61 14
##      1  6 46
ConfusionTable(ct)

## $Accuracy
## [1] 0.8425197
##
## $`True Positive Rate`
## [1] 0.7666667
##
## $`False Posistive Rate`
## [1] 0.08955224
##
## $`Total Error Rate`
## [1] 0.1574803
```

The result is pretty good, with error rate=15.75%, true positive Rate=76.67%, and false positive=8.96%. Then we want remove the not so significant varibales *zn*, *tax*, *black*, *medv* and refit the model.

```
lda.fit <- lda(crim01 ~ nox + dis + rad + ptratio, data=Boston.train)
lda.pred=predict(lda.fit,Boston.test)$class
ct <- table(lda.pred,Boston.test$crim01)
ct

##
## lda.pred 0 1
##      0 61 15
##      1  6 45
ConfusionTable(ct)

## $Accuracy
## [1] 0.8346457
##
## $`True Positive Rate`
## [1] 0.75
##
## $`False Posistive Rate`
## [1] 0.08955224
##
## $`Total Error Rate`
## [1] 0.1653543
```

The model does not go worse so much, so the the model with four variables may be enough. If we continue to remove variable, for example *rad*, the result is like

```

## 
## lda.pred 0 1
##          0 57 9
##          1 10 51

## $Accuracy
## [1] 0.8503937
##
## $`True Positive Rate`
## [1] 0.85
##
## $`False Posistive Rate`
## [1] 0.1492537
##
## $`Total Error Rate`
## [1] 0.1496063

```

The overall performance becomes better, but what is more important is the true positive rate is greatly improved, since in the problem we do not want misclassify a town with high crime rate into the group of low crime rate, which is very dangerous. So we may prefer more a model with high true positive rate than other performance indices. I also try some other changes, but none of them can perform better, so this may be the best model for LDA.

I will use similar steps above to compare the logistic regression model. Also, I fit the model with seven variables and four variables,

```

logis.fit <- glm(crim01 ~ zn + nox + black + dis + rad + tax + ptratio + medv, data=Boston.train, family = binomial)
logis.probs<-predict(logis.fit, Boston.test, type="response")
logis.pred <- rep(0, nrow(Boston.test))
logis.pred[logis.probs > 0.50] <- 1
ct <- table(logis.pred,Boston.test$crim01)
ct

## 
## logis.pred 0 1
##          0 59 8
##          1 8 52

ConfusionTable(ct)

## $Accuracy
## [1] 0.8740157
##
## $`True Positive Rate`
## [1] 0.8666667
##
## $`False Posistive Rate`
## [1] 0.119403
##
## $`Total Error Rate`
## [1] 0.1259843

logis.fit <- glm(crim01 ~ nox + dis + rad + ptratio, data=Boston.train, family = binomial)
logis.probs <- predict(logis.fit, Boston.test, type="response")
logis.pred <- rep(0, nrow(Boston.test))
logis.pred[logis.probs > 0.50] <- 1
ct <- table(logis.pred,Boston.test$crim01)
ct

```

```

## 
## logis.pred 0 1
##          0 63 12
##          1  4 48
ConfusionTable(ct)

## $Accuracy
## [1] 0.8740157
##
## $`True Positive Rate`
## [1] 0.8
##
## $`False Posistive Rate`
## [1] 0.05970149
##
## $`Total Error Rate`
## [1] 0.1259843

```

The model with seven predictors performs a little better than the one with four predictors, and also a little better compared to the LDA with four predictors. It is notable that if we fit a logistic regression model with the four predictor, but use the interaction $nox * dis$ instead of the predictors themselves, we can get similar result like the model with seven predictos. So sometimes maybe we can use a nonlinear boundary in low dimension to replace the linear boundary in high dimension without reduce the performance, but it becomes easier to interpret with less predictors.

Because of the previous attempts, we should believe that the four variables $-nox$, dis , rad , $ptratio$ is enough to make a good prediction. Then we use these predictors to fit the KNN models with differen K .

```

set.seed(1)
Boston.train.knn <- Boston.train[,c("nox", "dis", "rad", "ptratio")]
Boston.test.knn <- Boston.test[,c("nox", "dis", "rad", "ptratio")]
knn.pred <- knn(Boston.train.knn, Boston.test.knn, Boston.train$crim01, k=1)
ct <- table(knn.pred, Boston.test$crim01)
ct

## 
## knn.pred 0 1
##          0 64 1
##          1  3 59
ConfusionTable(ct)

## $Accuracy
## [1] 0.9685039
##
## $`True Positive Rate`
## [1] 0.9833333
##
## $`False Posistive Rate`
## [1] 0.04477612
##
## $`Total Error Rate`
## [1] 0.03149606

knn.pred=knn(Boston.train.knn, Boston.test.knn, Boston.train$crim01, k=2)
ct <- table(knn.pred, Boston.test$crim01)
ct

```

```
##  
## knn.pred 0 1  
##          0 63 3  
##          1 4 57  
ConfusionTable(ct)  
  
## $Accuracy  
## [1] 0.9448819  
##  
## $`True Positive Rate`  
## [1] 0.95  
##  
## $`False Posistive Rate`  
## [1] 0.05970149  
##  
## $`Total Error Rate`  
## [1] 0.05511811  
knn.pred=knn(Boston.train.knn,Boston.test.knn,Boston.train$crim01,k=3)  
ct <- table(knn.pred, Boston.test$crim01)  
ct  
  
##  
## knn.pred 0 1  
##          0 64 3  
##          1 3 57  
ConfusionTable(ct)  
  
## $Accuracy  
## [1] 0.9527559  
##  
## $`True Positive Rate`  
## [1] 0.95  
##  
## $`False Posistive Rate`  
## [1] 0.04477612  
##  
## $`Total Error Rate`  
## [1] 0.04724409  
knn.pred=knn(Boston.train.knn,Boston.test.knn,Boston.train$crim01,k=4)  
ct <- table(knn.pred, Boston.test$crim01)  
ct  
  
##  
## knn.pred 0 1  
##          0 65 3  
##          1 2 57  
ConfusionTable(ct)  
  
## $Accuracy  
## [1] 0.9606299  
##  
## $`True Positive Rate`  
## [1] 0.95
```

```

##  

## $`False Posistive Rate`  

## [1] 0.02985075  

##  

## $`Total Error Rate`  

## [1] 0.03937008  

knn.pred=knn(Boston.train.knn,Boston.test.knn,Boston.train$crim01,k=5)  

ct <- table(knn.pred, Boston.test$crim01)  

ct

##  

## knn.pred 0 1  

##          0 63 3  

##          1  4 57  

ConfusionTable(ct)

## $Accuracy  

## [1] 0.9448819  

##  

## $`True Positive Rate`  

## [1] 0.95  

##  

## $`False Posistive Rate`  

## [1] 0.05970149  

##  

## $`Total Error Rate`  

## [1] 0.05511811  

knn.pred=knn(Boston.train.knn,Boston.test.knn,Boston.train$crim01,k=11)  

ct <- table(knn.pred, Boston.test$crim01)  

ct

##  

## knn.pred 0 1  

##          0 61 3  

##          1  6 57  

ConfusionTable(ct)

## $Accuracy  

## [1] 0.9291339  

##  

## $`True Positive Rate`  

## [1] 0.95  

##  

## $`False Posistive Rate`  

## [1] 0.08955224  

##  

## $`Total Error Rate`  

## [1] 0.07086614

```

We can see that when $K = 1$, the true positive rate and the accuracy are both largest. Furthermore it is much better than the other two kinds of models. We can infer that the decision boundary is very likely to be nonlinear, even more complicated than quadratic form.

► Appendices

Code of function ConfusionTable()

```
ConfusionTable <- function(ct){  
    accuracy <- (ct[1, 1] + ct[2, 2]) / (sum(ct))  
    TPr <- ct[2, 2] / (ct[1, 2] + ct[2, 2])  
    FPr <- ct[2, 1] / (ct[1, 1] + ct[2, 1])  
    precision <- ct[2, 2] / (ct[2, 1] + ct[2, 2])  
    error <- 1 - accuracy  
    result <- list(accuracy, TPr, FPr, precision, error)  
    names(result) <- c("Accuracy", "True Positive Rate",  
                      "False Posistive Rate", "Precision",  
                      "Total Error Rate")  
    return(result)  
}
```