**Chapter 4: Classification**

**1**



**2**

设为一定值，对公式(4.12)取对数得到



因为是定值，也为定值，将其移到等式左边得到



此即为公式(4.13)，即将一个观测分入使(4.12)最大的一类与将一个观测分入使(4.13)最大的一类是等价的。

**3**

对于QDA，p=1时，

设为一定值，两边取对数得到



因为是定值，也是定值，但不再是定值了，移项得到



贝叶斯分类器不再是线性的，而是二次的。

**4**

**(a)**

均匀分布，10%。

**(b)**

易知1%。

**(c)**

易知。

**(d)**

维数每增1，预测模型需要观测中数据的比例减小为原来的10%。

**(e)**

设边长为。

，。

时，。

时，。

时，。

**5**

**(a)**

对于training set，QDA表现更好，因为相比于LDA，QDA更加flexible，更接近训练集的数据，对训练集拟合得更好。对于test set，LDA表现更好，因为原本贝叶斯决策边界就是线性的，LDA估计得会更精确，QDA会出现过拟合的问题。

**(b)**

对于training set和test set，都是QDA表现会更好。

**(c)**

QDA预测的准确度会提高，因为随着观测数据的增加，一个更flexible的模型对数据拟合得会更好，方差大的问题也会因数据量足够大而相抵消。

**(d)**

False。如果训练集的数据量相对于维数来说不是很大的话，因为QDA相对于LDA更加flexible，前者拟合得效果会比较差，会出现过拟合的问题，造成test error rate比较大。

**6**

**(a)**



代入得到

。

**(b)**

由，得到

。

**7**



设发放股利为类1，不发放股利为类0。代入数值得到



代入，得到



**8**

K=1，KNN的training error rate为0，所以其test error rate为36%，比logistic回归的test error rate高，所以用logistic回归更好。

**9**

**(a)**



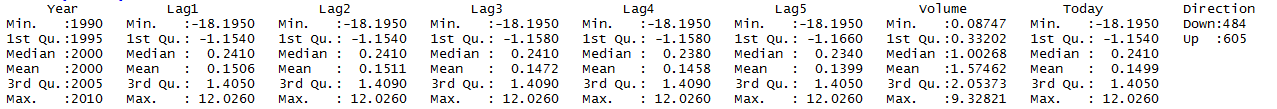
得到。

**(b)**

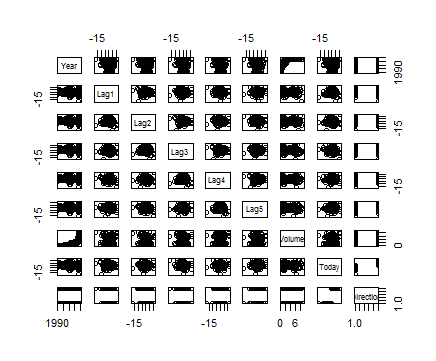
。

**10(a)**

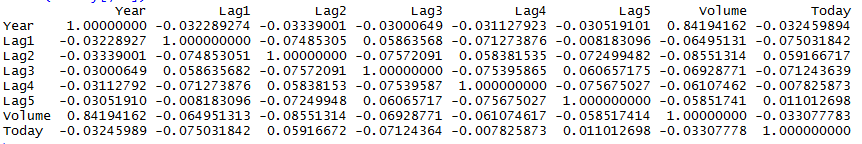
> summary(Weekly)



> pairs(Weekly)



> cor(Weekly[,-9])



Year和Volume具有相关性，其余变量之间不明显。

**10(b)**

> attach(Weekly)

> glm.fit10<-glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume, data = Weekly, family = binomial)

> summary(glm.fit10)

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在0.01的水平上是显著的。

**10(c)**

> glm.fit10probs = predict(glm.fit10, type = "response")

> glm.fit10pred = rep("Down",length(glm.fit10probs))

> glm.fit10pred[glm.fit10probs>0.5] = "Up"

> table(glm.fit10pred,Direction)

Direction

glm.fit10pred Down Up

Down 54 48

Up 430 557

总体正确率为(54+557)/(54+557+48+430) = 56.1%。正确预测为“Down”的比例为54/(430+54) = 11.2%，正确预测为“Up”的比例为557/(557+48) = 92.1%。

**10(d)**

> train = (Year < 2009)

> Weekly.0910 = Weekly[!train,]

> Direction.0910 = Direction[!train]

> glm.fit10\_ = glm(Direction~Lag2,data = Weekly,family = binomial,subset = train)

> glm.probs\_ = predict(glm.fit10\_,Weekly.0910,type = "response")

> glm.pred\_ = rep("Down",length(glm.probs\_))

> glm.pred\_[glm.probs\_ >0.5] = "Up"

> table(glm.pred\_,Direction.0910)

Direction.0910

glm.pred\_ Down Up

Down 9 5

Up 34 56

> mean(glm.pred\_ == Direction.0910)

[1] 0.625

**10(e)**

> lda.fit10 = lda(Direction ~ Lag2, data = Weekly, subset = train)

> lda.pred = predict(lda.fit10, Weekly.0910)

> table(lda.pred$class, Direction.0910)

Direction.0910

Down Up

Down 9 5

Up 34 56

> mean(lda.pred$class == Direction.0910)

[1] 0.625

**10(f)**

> qda.fit10 = qda(Direction ~ Lag2, data = Weekly, subset = train)

> qda.pred = predict(qda.fit10, Weekly.0910)

> table(qda.pred$class, Direction.0910)

Direction.0910

Down Up

Down 0 0

Up 43 61

> mean(qda.pred$class == Direction.0910)

[1] 0.5865385

**10(g)**

> train.X = as.matrix(Lag2[train])

> test.X = as.matrix(Lag2[!train])

> train.Direction = Direction[train]

> set.seed(1)

> knn.pred = knn(train.X, test.X, train.Direction, k = 1)

> table(knn.pred, Direction.0910)

Direction.0910

knn.pred Down Up

Down 21 30

Up 22 31

> mean(knn.pred == Direction.0910)

[1] 0.5

**10(h)**

Logistic回归和LDA效果更好。QDA预测正确预测为“Up”的比例为100%，但是正确预测为“Down”的比例为0，KNN with K=1效果不太好。

**10(i)**

**Variable：Lag1:Lag2**

**Logistics Regression**

> train = (Year < 2009)

> Weekly.0910 = Weekly[!train,]

> Direction.0910 = Direction[!train]

> glm.fit10 = glm(Direction~Lag1:Lag2,data = Weekly,family = binomial,subset = train)

> glm.probs = predict(glm.fit10,Weekly.0910,type = "response")

> glm.pred = rep("Down",length(glm.probs))

> glm.pred[glm.probs >0.5] = "Up"

> table(glm.pred,Direction.0910)

Direction.0910

glm.pred Down Up

Down 1 1

Up 42 60

> mean(glm.pred == Direction.0910)

[1] 0.5865385

**LDA**

> lda.fit10 = lda(Direction ~Lag1:Lag2, data = Weekly, subset = train)

> lda.pred = predict(lda.fit10, Weekly.0910)

> table(lda.pred$class, Direction.0910)

Direction.0910

Down Up

Down 0 1

Up 43 60

> mean(lda.pred$class == Direction.0910)

[1] 0.5769231

**QDA**

> qda.fit10 = qda(Direction ~ Lag1:Lag2, data = Weekly, subset = train)

> qda.pred = predict(qda.fit10, Weekly.0910)

> table(qda.pred$class, Direction.0910)

Direction.0910

Down Up

Down 16 32

Up 27 29

> mean(qda.pred$class == Direction.0910)

[1] 0.4326923

**KNN，K=1**

> Lag12=Lag1\*Lag2

> train.X = as.matrix(Lag12[train])

> test.X = as.matrix(Lag12[!train])

> train.Direction = Direction[train]

> set.seed(1)

> knn.pred = knn(train.X, test.X, train.Direction, k = 1)

> table(knn.pred, Direction.0910)

Direction.0910

knn.pred Down Up

Down 18 33

Up 25 28

> mean(knn.pred == Direction.0910)

[1] 0.4423077

**KNN，K=10**

> set.seed(1)

> knn.pred = knn(train.X, test.X, train.Direction, k = 10)

> table(knn.pred, Direction.0910)

Direction.0910

knn.pred Down Up

Down 16 27

Up 27 34

> mean(knn.pred == Direction.0910)

[1] 0.4807692

**KNN，K=100**

> train.Direction = Direction[train]

> set.seed(1)

> knn.pred = knn(train.X, test.X, train.Direction, k = 100)

> table(knn.pred, Direction.0910)

Direction.0910

knn.pred Down Up

Down 5 6

Up 38 55

> mean(knn.pred == Direction.0910)

[1] 0.5769231

**Variable：Lag1\*Lag2**

**Logistics Regression**

> glm.fit10 = glm(Direction~Lag1\*Lag2,data = Weekly,family = binomial,subset = train)

> glm.probs = predict(glm.fit10,Weekly.0910,type = "response")

> glm.pred = rep("Down",length(glm.probs))

> glm.pred[glm.probs >0.5] = "Up"

> table(glm.pred,Direction.0910)

Direction.0910

glm.pred Down Up

Down 7 8

Up 36 53

> mean(glm.pred == Direction.0910)

[1] 0.5769231

**LDA**

> lda.fit10 = lda(Direction ~Lag1\*Lag2, data = Weekly, subset = train)

> lda.pred = predict(lda.fit10, Weekly.0910)

> table(lda.pred$class, Direction.0910)

Direction.0910

Down Up

Down 7 8

Up 36 53

> mean(lda.pred$class == Direction.0910)

[1] 0.5769231

**QDA**

> qda.fit10 = qda(Direction ~ Lag1\*Lag2, data = Weekly, subset = train)

> qda.pred = predict(qda.fit10, Weekly.0910)

> table(qda.pred$class, Direction.0910)

Direction.0910

Down Up

Down 23 36

Up 20 25

> mean(qda.pred$class == Direction.0910)

[1] 0.4615385

**KNN，K=1**

> train.X = cbind(Lag1,Lag2,Lag12)[train,]

> test.X = cbind(Lag1,Lag2,Lag12)[!train,]

> train.Direction = Direction[train]

> set.seed(1)

> knn.pred = knn(train.X, test.X, train.Direction, k = 1)

> table(knn.pred, Direction.0910)

Direction.0910

knn.pred Down Up

Down 18 25

Up 25 36

> mean(knn.pred == Direction.0910)

[1] 0.5192308

**KNN，K=10**

> set.seed(1)

> knn.pred = knn(train.X, test.X, train.Direction, k = 10)

> table(knn.pred, Direction.0910)

Direction.0910

knn.pred Down Up

Down 20 25

Up 23 36

> mean(knn.pred == Direction.0910)

[1] 0.5384615

**KNN，K=100**

> set.seed(1)

> knn.pred = knn(train.X, test.X, train.Direction, k = 100)

> table(knn.pred, Direction.0910)

Direction.0910

knn.pred Down Up

Down 10 16

Up 33 45

> mean(knn.pred == Direction.0910)

[1] 0.5288462

并没有单独用Lag2做预测变量得到的结果好。

**11(a)**

> attach(Auto)

> mpg01 = rep(0, length(mpg))

> mpg01[mpg > median(mpg)] = 1

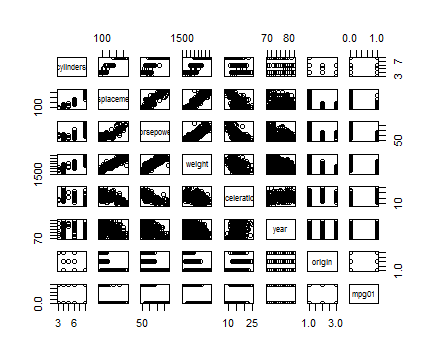
> Auto = data.frame(Auto, mpg01)

**11(b)**

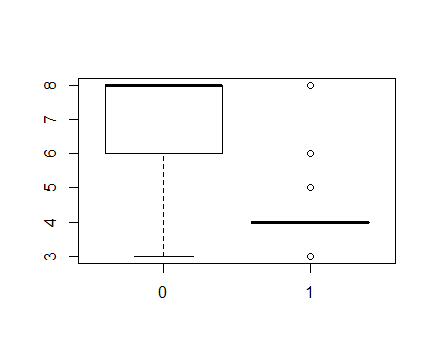
> Auto\_ = Auto[,-1]

> Auto\_ = Auto\_[,-8]

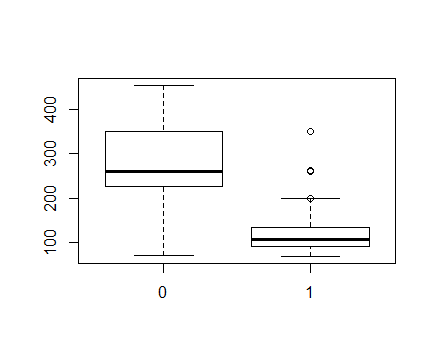
> pairs(Auto\_)



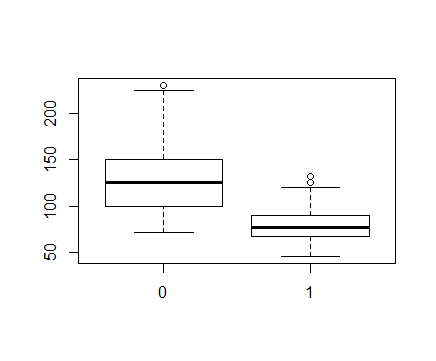
> boxplot(cylinders~mpg01)



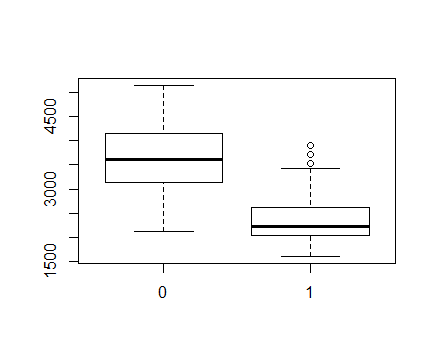
> boxplot(displacement~mpg01)



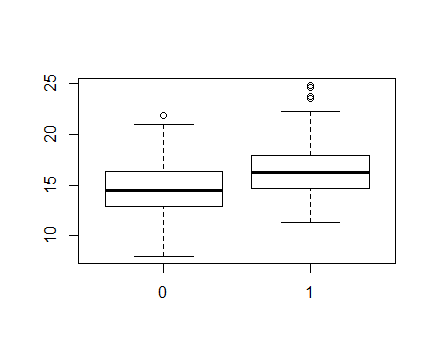
> boxplot(horsepower~mpg01)



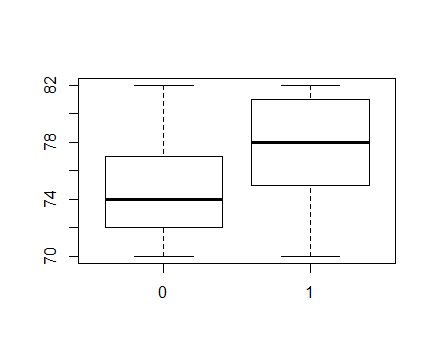
> boxplot(weight~mpg01)



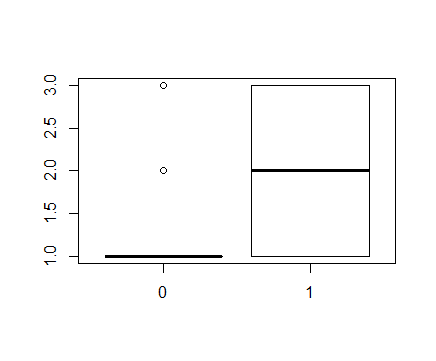
> boxplot(acceleration~mpg01)



> boxplot(year~mpg01)



> boxplot(origin~mpg01)



由箱线图可以看出，mpg01和cylinders，displacement，horsepower，weight是相关的。

**11(c)**

> train = (year%%2 == 0)

> test = !train

> Auto.train = Auto[train,]

> Auto.test = Auto[test,]

> mpg01.test = mpg01[test]

**11(d)**

> lda.fit11 = lda(mpg01 ~ cylinders + weight + displacement + horsepower, data = Auto, subset = train)

> lda.pred = predict(lda.fit11, Auto.test)

> mean(lda.pred$class != mpg01.test)

[1] 0.1263736

Test error rate 为0.1263736。

**11(e)**

> qda.fit11 = qda(mpg01 ~ cylinders + weight + displacement + horsepower, data = Auto, subset = train)

> qda.pred = predict(qda.fit11, Auto.test)

> mean(qda.pred$class != mpg01.test)

[1] 0.1318681

Test error rate 为0.1318681。

**11(f)**

> glm.fit11 = glm(mpg01 ~ cylinders + weight + displacement + horsepower, data = Auto,

+ family = binomial, subset = train)

> glm.probs = predict(glm.fit11, Auto.test, type = "response")

> glm.pred = rep(0, length(glm.probs))

> glm.pred[glm.probs > 0.5] = 1

> mean(glm.pred != mpg01.test)

[1] 0.1208791

Test error rate 为0.1208791

**11(g)**

> library(class)

> train.X = cbind(cylinders, weight, displacement, horsepower)[train, ]

> test.X = cbind(cylinders, weight, displacement, horsepower)[test, ]

> train.mpg01 = mpg01[train]

> set.seed(1)

> knn.pred = knn(train.X, test.X, train.mpg01, k = 1)

> mean(knn.pred != mpg01.test)

[1] 0.1538462

> knn.pred = knn(train.X, test.X, train.mpg01, k = 10)

> mean(knn.pred != mpg01.test)

[1] 0.1648352

> knn.pred = knn(train.X, test.X, train.mpg01, k = 100)

> mean(knn.pred != mpg01.test)

[1] 0.1428571

K取1,10和100时，Test error rate 分别为0.1538462，0.1648352，0.1428571。

**12(a)**

> Power=function(){

+ print(2^3)

+ }

> Power()

[1] 8

**12(b)**

> Power=function(){

+ print(2^3)

+ }

>

> Power()

[1] 8

> Power2=function(x,a){

+ print(x^a)

+ }

> Power2(3,8)

[1] 6561

**12(c)**

> Power2(10,3)

[1] 1000

> Power2(8,17)

[1] 2.2518e+15

> Power2(131,3)

[1] 2248091

**12(d)**

> Power3=function(x,a){

+ result=x^a

+ return(result)

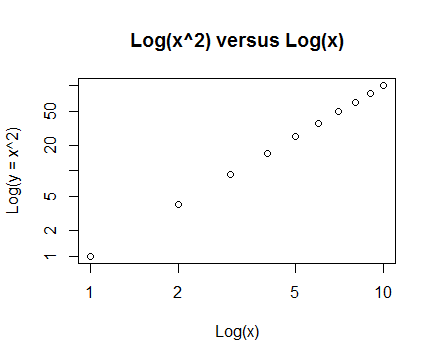
+ }

**12(e)**

> x = 1:10

> plot(x, Power3(x, 2), log = "xy", xlab = "Log(x)", ylab = "Log(y = x^2)",

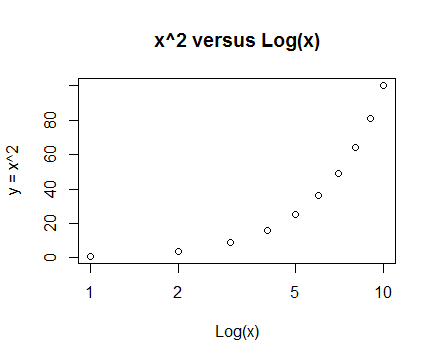
+ main = "Log(x^2) versus Log(x)")



> x = 1:10

> plot(x, Power3(x, 2), log = "x", xlab = "Log(x)", ylab = "y = x^2",

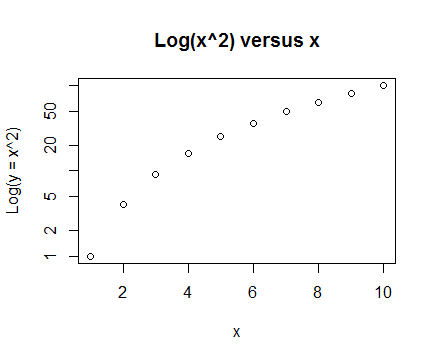
+ main = "x^2 versus Log(x)")



> x = 1:10

> plot(x, Power3(x, 2), log = "y", xlab = "x", ylab = "Log(y = x^2)",

+ main = "Log(x^2) versus x")



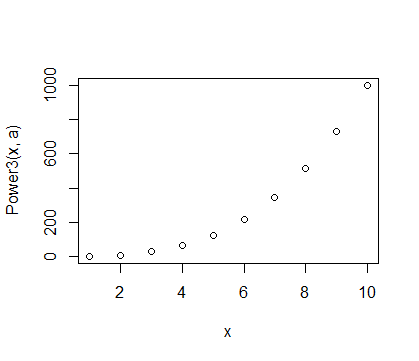
**12(f)**

> PlotPower=function(x,a){

+ plot(x,Power3(x,a))

+ }

> PlotPower(1:10,3)



**13**

> crime01 = rep(0, length(crim))

> crime01[crim > median(crim)] = 1

> Boston = data.frame(Boston, crime01)

> train = 1:(dim(Boston)[1]/2)

> test = (dim(Boston)[1]/2 + 1):dim(Boston)[1]

> Boston.train = Boston[train, ]

> Boston.test = Boston[test, ]

> crime01.test = crime01[test]

> glm.fit13 = glm(crime01 ~ . -chas - crime01 - crim, data = Boston, family = binomial, subset = train)

> glm.probs = predict(glm.fit13, Boston.test, type = "response")

> glm.pred = rep(0, length(glm.probs))

> glm.pred[glm.probs > 0.5] = 1

> mean(glm.pred != crime01.test)

[1] 0.1818182

test error rate为0.1818182。

> glm.fit13\_ = glm(crime01 ~ . - tax -chas - crime01 - crim, data = Boston, family = binomial, subset = train)

> glm.probs = predict(glm.fit13\_, Boston.test, type = "response")

> glm.pred = rep(0, length(glm.probs))

> glm.pred[glm.probs > 0.5] = 1

> mean(glm.pred != crime01.test)

[1] 0.1857708

test error rate为0.1857708。

> glm.fit13\_\_ = glm(crime01 ~ . -zn - indus - ptratio -chas - crime01 - crim, data = Boston, family = binomial, subset = train)

> glm.probs = predict(glm.fit13\_\_, Boston.test, type = "response")

> glm.pred = rep(0, length(glm.probs))

> glm.pred[glm.probs > 0.5] = 1

> mean(glm.pred != crime01.test)

[1] 0.09486166

test error rate为0.09486166。

> lda.fit13 = lda(crime01 ~ . -chas - crime01 - crim, data = Boston, subset = train)

> lda.pred = predict(lda.fit13, Boston.test)

> mean(lda.pred$class != crime01.test)

[1] 0.1225296

test error rate为0.1225296。

> lda.fit13\_ = lda(crime01 ~ . - tax -chas - crime01 - crim, data = Boston, subset = train)

> lda.pred = predict(lda.fit13\_, Boston.test)

> mean(lda.pred$class != crime01.test)

[1] 0.1225296

test error rate为0.1225296。

> lda.fit13\_\_ = lda(crime01 ~ . -zn - indus - ptratio -chas - crime01 - crim, data = Boston, subset = train)

> lda.pred = predict(lda.fit13\_\_, Boston.test)

> mean(lda.pred$class != crime01.test)

[1] 0.09881423

test error rate为0.09881423。

> 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.crime01 = crime01[train]

> set.seed(1)

> knn.pred = knn(train.X, test.X, train.crime01, k = 1)

> mean(knn.pred != crime01.test)

[1] 0.458498

test error rate为0.458498。

> knn.pred = knn(train.X, test.X, train.crime01, k = 10)

> mean(knn.pred != crime01.test)

[1] 0.1185771

test error rate为0.1185771。

> knn.pred = knn(train.X, test.X, train.crime01, k = 100)

> mean(knn.pred != crime01.test)

[1] 0.4901186

test error rate为0.4901186。

> train.X = cbind(nox, rm, dis, rad, black, medv)[train, ]

> test.X = cbind(nox, rm, dis, rad, black, medv)[test, ]

> knn.pred = knn(train.X, test.X, train.crime01, k = 10)

> mean(knn.pred != crime01.test)

[1] 0.3241107

test error rate为0.3241107。