Classification and Methods for Modeling

Jorge Celaya and Austin Lapenna

Contents

Chapter 5 2
5. 2
8. 2
Chapter 6 4
8. 4
9. 9
10. 10
Code Index 12
Chapter 4 12
10. 12
12. 15
Chapter 5 15
5. 15
8. 16
Chapter 6 17
8. 17
9. 20
11. 21

Chapter 5

5.

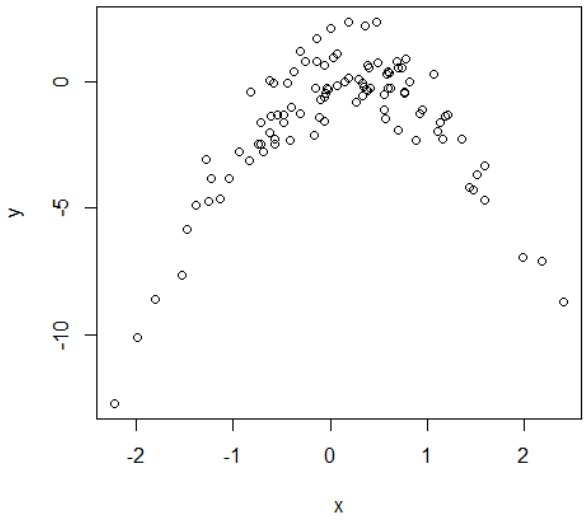
a. From our logistic regression model, we obtain very small coefficient estimates, which tells us that there is a small probability that a person will default.

```
> glm.fit

Call:  glm(formula = default ~ income + balance, family = binomial,
    data = Default)

Coefficients:
(Intercept)       income      balance
 -1.154e+01    2.081e-05    5.647e-03

Degrees of Freedom: 9999 Total (i.e. Null);  9997 Residual
Null Deviance:      2921
Residual Deviance: 1579         AIC: 1585
```

b. The estimated test error from the validation set approach is 0.0262, so only about 2.5% of the predictions were wrong.
c. The validation test error ranged from around 0.025 to 0.037, so our model seems to be performing consistently well.
d. Including a dummy variable for students in our logistic regression model results in a validation test error of 0.0272, which is in the same range as before. Hence, there appears to be no reduction in the test error rate with this addition to the model.

8.

a) In this data set, n = 100 and p = 2. The equation follows the form $Y = X - X^2 + \varepsilon$.
b) It looks like Y is a parabolic function of X.



c) The LOOCV errors obtained from the polynomial models of degree one, two, three, and four are 7.288, 0.937, 0,957, 0.954, respectively.

d) Using another seed resulted in the same LOOCV errors, which is the expected behavior because LOOCV averages the errors from leaving each individual observation out.

e) The model with the smallest LOOCV error was unsurprisingly the quadratic model because it most closely resembles the true underlying function. The linear model performed the worst and the higher degree polynomial models performed fairly well, but not as good as the quadratic model.

f) The coefficient estimates for β_0, β_1, and β_2 had the highest statistical significance, while the estimates for β_3 and β_4 hold much lower statistical significance. These results are expected, especially because the coefficients for $\beta_3$and β_4 do not actually exist in the underlying model. Consequently, the cross-validation errors for those models which contain only statistically significant coefficient estimates perform the best.

```
> summary(glm.fit)

Call:
glm(formula = y ~ poly(x, 4))

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.0550  -0.6212  -0.1567   0.5952   2.2267

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  -1.55002    0.09591 -16.162  < 2e-16 ***
poly(x, 4)1   6.18883    0.95905   6.453 4.59e-09 ***
poly(x, 4)2 -23.94830    0.95905 -24.971  < 2e-16 ***
poly(x, 4)3   0.26411    0.95905   0.275    0.784
poly(x, 4)4   1.25710    0.95905   1.311    0.193
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 0.9197797)

    Null deviance: 700.852  on 99  degrees of freedom
Residual deviance:  87.379  on 95  degrees of freedom
AIC: 282.3

Number of Fisher Scoring iterations: 2
```

Chapter 6

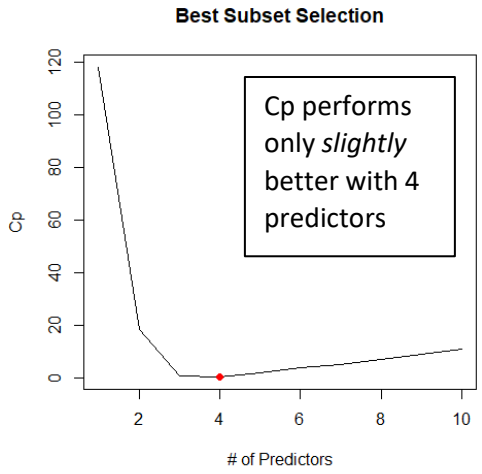
8.

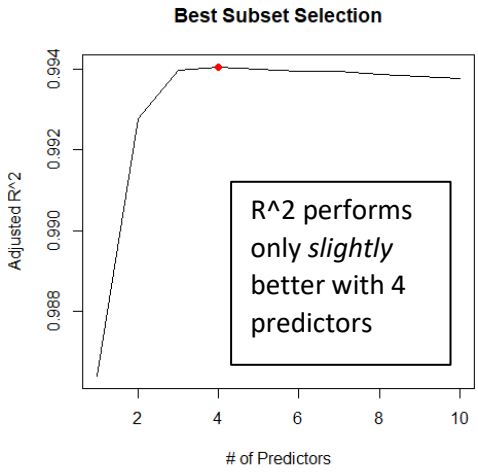
a) We create vectors X and eps both of size 100:

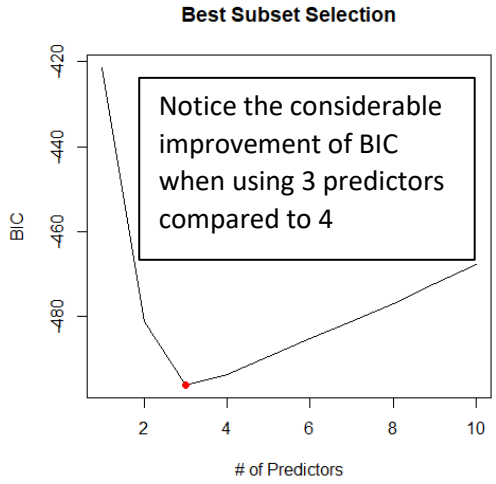
```
> summary(X)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-2.2147 -0.4942  0.1139  0.1089  0.6915  2.4016
> summary(eps)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-1.91436 -0.65105 -0.17722 -0.03781  0.50090  2.30798
```

b) We model Y using the coefficients $\beta_0 = 2$, $\beta_1 = 2$, $\beta_2 = 0.5$, $\beta_3 = 4$ to obtain $Y = 2 + 2X + 0.5X^2 + 4X^3 + \epsilon$.

c) Our results show us that on average, the best model contains 4 predictors, except for the BIC criterion, which performed best with 3 predictors.



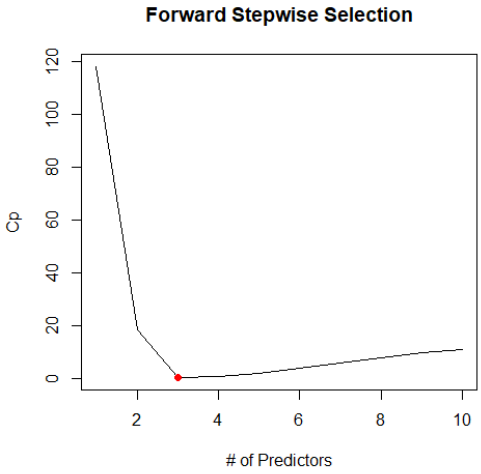


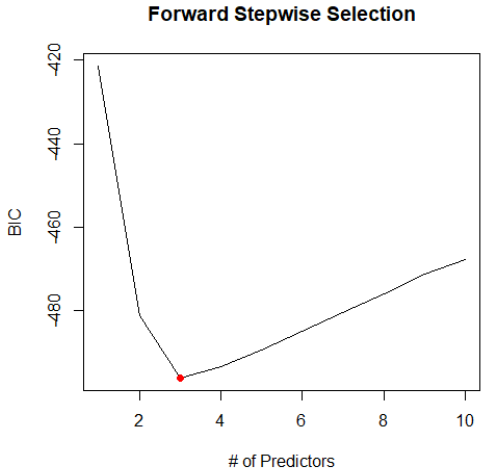


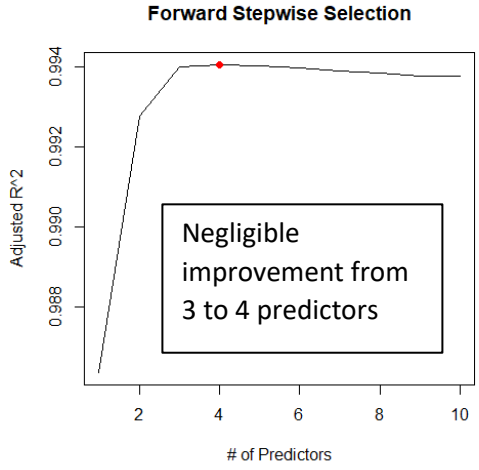
It is important to note that the 4-predictor model improvements for C_p and adjusted R^2 are marginal and perform similarly to the 3-predictor model. Therefore, we chose the 3-predictor model for better interpretability and similar performance. We obtain the following predictor coefficients from best subset selection: $\beta_1 \approx 2.043$, $\beta_3 \approx 3.985$, $\beta_4 \approx 0.888$.

```
> coefficients(regfit.full, id = 3)
          (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)3 poly(x, 10, raw = T)4
           2.19243917          2.04334443          3.98543484          0.08876886
```

d) In the case of forward stepwise selection, we obtain the best results for C_p and BIC when we use 3 predictors, while adjusted R^2's improvement from the 3-predictor model to the 4-predictor model is miniscule. This leads us to choosing the 3-variable model again for forward stepwise selection.



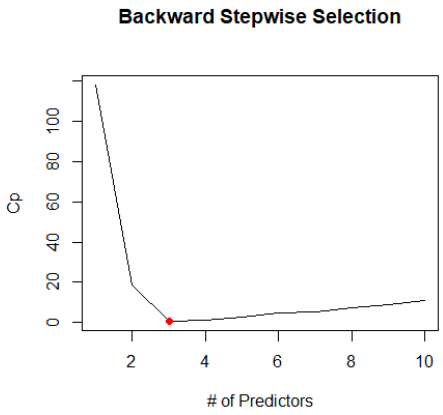


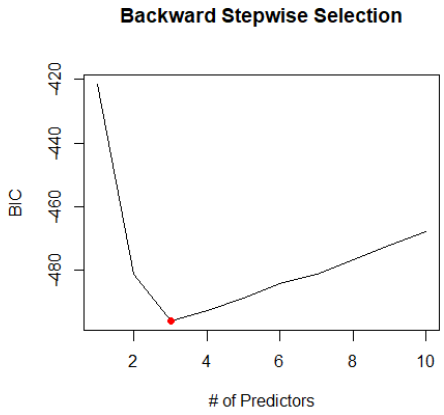


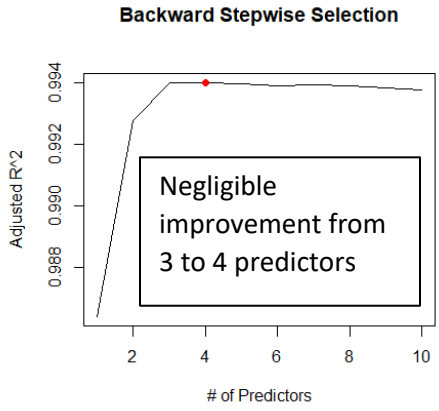
Forward stepwise leads us to the same predictor coefficients as the best subset selection model: $\beta_1 \approx 2.043$, $\beta_3 \approx 3.985$, $\beta_4 \approx 0.888$.

```
> coefficients(regfit.fwd, id = 3)
          (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)3 poly(x, 10, raw = T)4
           2.19243917          2.04334443          3.98543484          0.08876886
```

We get similar results for backward stepwise selection:



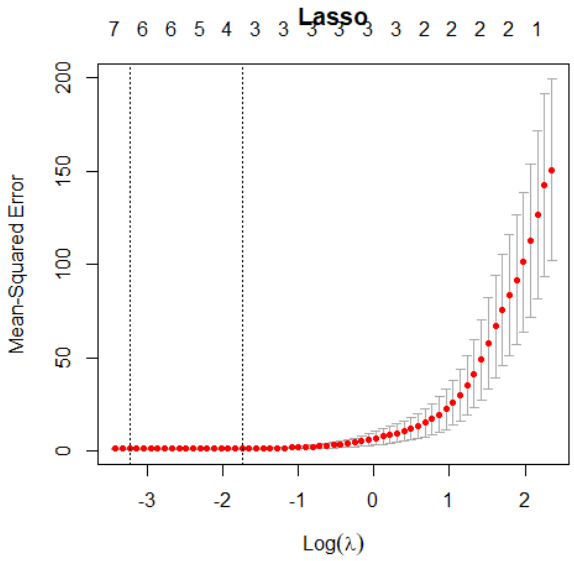




Along with the same predictor coefficients:

```
> coefficients(regfit.bwd, id = 3)
          (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)3 poly(x, 10, raw = T)4
           2.19243917          2.04334443          3.98543484          0.08876886
```

e) Using the lasso model, we find that the optimal value for λ is 0.0552 and we get the following graph:



From the lasso, we get coefficient estimates: $\beta_0 \approx 2.171$, $\beta_1 \approx 2.173$, $\beta_2 \approx 0.131$, $\beta_3 \approx 3.793$, $\beta_4 \approx 0.044$, $\beta_5 \approx 0.165$, and $\beta_7 \approx 0.004$.

```
> predict(best.model, s = best.lamda, type = "coefficients")
11 x 1 sparse Matrix of class "dgCMatrix"
                              s1
(Intercept)           2.171052149
poly(x, 10, raw = T)1 2.172838642
poly(x, 10, raw = T)2 0.130750135
poly(x, 10, raw = T)3 3.793103367
poly(x, 10, raw = T)4 0.043827920
poly(x, 10, raw = T)5 0.016457429
poly(x, 10, raw = T)6 .
poly(x, 10, raw = T)7 0.003817953
poly(x, 10, raw = T)8 .
poly(x, 10, raw = T)9 .
poly(x, 10, raw = T)10 .
```

f) We now look at the model $Y = \beta_0 + \beta_7 + \varepsilon$ and use the previous model selection methods. The choices for the number of predictors from the best subset selection in terms of C_p, BIC and adjusted R^2 vary quite a bit with values 1, 2, and 4, respectively. In any case, we see that a single-predictor model containing only $\beta_7 = 7.00077$ (and an intercept) is closest to predicting our underlying model, where $\beta_7 = 7$.

```
> coefficients(regfit.full, id = 1)
         (Intercept) poly(x, 10, raw = T)7
             1.95894              7.00077
> coefficients(regfit.full, id = 2)
         (Intercept) poly(x, 10, raw = T)2 poly(x, 10, raw = T)7
           2.0704904           -0.1417084            7.0015552
> coefficients(regfit.full, id = 4)
         (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2 poly(x, 10, raw = T)3 poly(x, 10, raw = T)7
           2.0762524            0.2914016           -0.1617671           -0.2526527            7.0091338
```

Additionally, we see that the chosen model from the lasso is the single-variable model containing $\beta_7 = 6.796694$, but we note that these estimates are further from the underlying model than the single-variable model obtained from best subset selection.

```
> predict(best.model, s = best.lamda, type = "coefficients")
11 x 1 sparse Matrix of class "dgCMatrix"
                              s1
(Intercept)            2.820215
poly(x, 10, raw = T)1  .
poly(x, 10, raw = T)2  .
poly(x, 10, raw = T)3  .
poly(x, 10, raw = T)4  .
poly(x, 10, raw = T)5  .
poly(x, 10, raw = T)6  .
poly(x, 10, raw = T)7  6.796694
poly(x, 10, raw = T)8  .
poly(x, 10, raw = T)9  .
poly(x, 10, raw = T)10 .
```

9.

g) Our training and test sets consist of a random subset of half the data.

College.test	389 obs. of 18 variables
College.train	388 obs. of 18 variables

h) We got a test MSE of 1,135,758 for our fitted linear model.

i) We got a test MSE of 976,261 for ridge regression using $\lambda = 405.8404$ and 17 non-zero coefficient estimates.

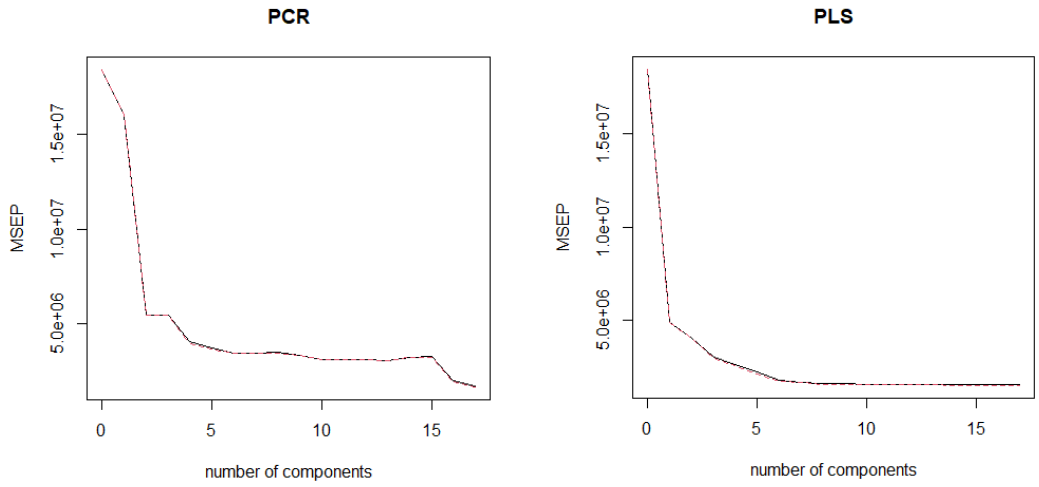
```
> ridge.mod

Call:  cv.glmnet(x = train.mat, y = College.train$Apps, alpha = 0)

Measure: Mean-Squared Error

    Lambda Index Measure      SE Nonzero
min  405.8   100 2395532 1265069      17
1se 2608.8    80 3658605 2187953      17
```

j) We got a test MSE of 1,115,901 the lasso using $\lambda \approx 2$ and 17 non-zero coefficient estimates.

```
> lasso.mod

Call:  cv.glmnet(x = train.mat, y = College.train$Apps, alpha = 1)

Measure: Mean-Squared Error

    Lambda Index Measure      SE Nonzero
min    2.0    83 1648718 418612      17
1se  435.2    25 2028391 615771       2
```

k) The PCR model gave us a test MSE of 1,723,100 using M = 16 components.

l) The PLS model gave very similar results of 1,066,991 for the test MSE using M = 15 components.

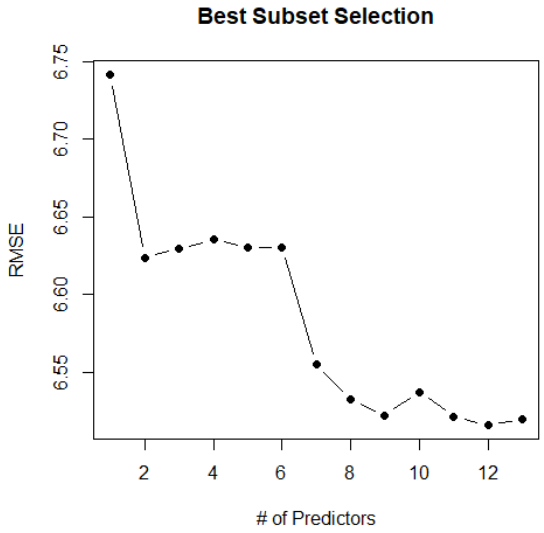


m) The linear, ridge regression, and lasso models all perform relatively close to each other in terms of testing accuracy, while PLS performs the worst by a considerable margin. With an average of 3,000 applications in the data set and a deviation of roughly ± 1,000 (33%), these models may be unsatisfactory if we require a small margin of error for expected prediction accuracy.

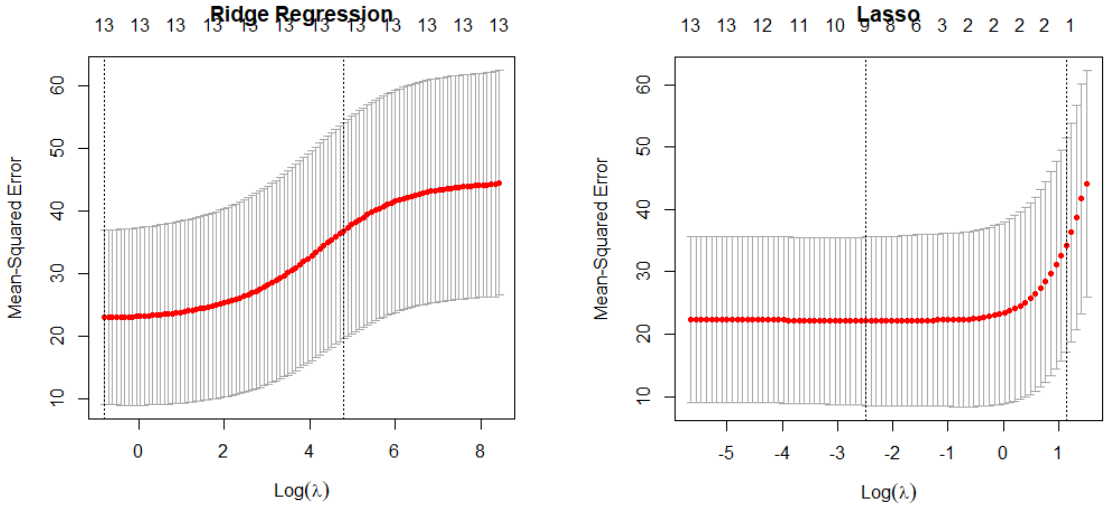
10.

a. From performing 10-fold cross-validation with best subset selection, ridge regression, the lasso, PCR, and PLS to model the per capital crime rate by town in the Boston data set, we get the following results.

We obtain a minimum RMSE of 6.516145 from our 10-fold cross-validation using best subset selection when the model contains 12 predictors. Note that this RMSE is relatively close to that of the 9-predictor best subset model.



From our ridge regression results, we obtained a test RMSE of 8.079883, which occurred when $\lambda = 0.5147957$. Better results emerged from our lasso model containing 2 predictors with a test RMSE of 7.743034.



Lastly, the PCR and PLS models gave cross-validation RMSE's of 6.599 and 6.563, respectively, both of which performed best with 13 components, indicating little improvement from increasing bias by decreasing dimensionality.

b. The model which performed the best in testing is the 10-fold cross-validated best subset selection model with 12 predictors, and the PCR and PLS models followed closely behind. Despite this, I would prefer the 9-predictor best subset selection model because it suffers only a minimal loss of accuracy and still outperforms PCR and PLS while being more interpretable.

c. The chosen best subset model contains 9 of the 13 features in the data set because it was shown that some of the predictors do not explain the responses in the data since we obtained a very similar RMSE when using 9 features instead of 13.

Code Index

Chapter 4

10.

```
library(ISLR)
library(class)


summary(Weekly)
pairs(Weekly)


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


glm.probs=predict(glm.fits,type="response")
contrasts(Direction)


glm.pred=rep("Down",1089)
glm.pred[glm.probs>.5]="Up"


table(glm.pred,Direction)
mean(glm.pred==Direction)


train=(Year<2009)
Weekly.0910=Weekly[!train,]
dim(Weekly.0910)
Direction.0910=Direction[!train]


glm.fits=glm(Direction~Lag2,data=Weekly,family=binomial,subset=train)
glm.probs=predict(glm.fits,Weekly.0910,type="response")
glm.pred=rep("Down",104)
```

```
glm.pred[glm.probs>.5]="Up"
table(glm.pred,Direction.0910)
mean(glm.pred==Direction.0910)


lda.fit=lda(Direction~Lag2,data=Weekly,subset=train)
lda.fit
lda.pred=predict(lda.fit,Weekly.0910)
lda.class=lda.pred$class
table(lda.class,Direction.0910)
mean(lda.class==Direction.0910)


lda.fit2=lda(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume,data=Weekly,subset=
train)
lda.fit2
lda.pred2=predict(lda.fit2,Weekly.0910)
lda.class2=lda.pred2$class
table(lda.class2,Direction.0910)
mean(lda.class2==Direction.0910)


lda.fit3=lda(Direction~Lag2+Lag3+Lag4+Lag5,data=Weekly,subset=train)
lda.fit3
lda.pred3=predict(lda.fit3,Weekly.0910)
lda.class3=lda.pred3$class
table(lda.class3,Direction.0910)
mean(lda.class3==Direction.0910)


lda.fit4=lda(Direction~Lag2+Lag5+Volume,data=Weekly,subset=train)
lda.fit4
lda.pred4=predict(lda.fit4,Weekly.0910)
lda.class4=lda.pred4$class
table(lda.class4,Direction.0910)
mean(lda.class4==Direction.0910)


qda.fit=qda(Direction~Lag2,data=Weekly,subset=train)
qda.fit
qda.class=predict(qda.fit,Weekly.0910)$class
table(qda.class,Direction.0910)
mean(qda.class==Direction.0910)
```

```
qda.fit2=qda(Direction~Lag1+Lag2+Volume,data=Weekly,subset=train)
qda.fit2
qda.class2=predict(qda.fit2,Weekly.0910)$class
table(qda.class2,Direction.0910)
mean(qda.class2==Direction.0910)


qda.fit3=qda(Direction~Lag1+Lag2+Lag3+Lag5+Volume,data=Weekly,subset=train
)
qda.fit3
qda.class3=predict(qda.fit3,Weekly.0910)$class
table(qda.class3,Direction.0910)
mean(qda.class3==Direction.0910)


qda.fit4=qda(Direction~Lag2+Volume,data=Weekly,subset=train)
qda.fit4
qda.class4=predict(qda.fit4,Weekly.0910)$class
table(qda.class4,Direction.0910)
mean(qda.class4==Direction.0910)


train.X=cbind(Lag1,Lag2)[train,]
test.X=cbind(Lag1,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)
mean(knn.pred==Direction.0910)


knn.pred=knn(train.X,test.X,train.Direction,k=2)
table(knn.pred,Direction.0910)
mean(knn.pred==Direction.0910)


knn.pred=knn(train.X,test.X,train.Direction,k=3)
table(knn.pred,Direction.0910)
mean(knn.pred==Direction.0910)


knn.pred=knn(train.X,test.X,train.Direction,k=4)
table(knn.pred,Direction.0910)
mean(knn.pred==Direction.0910)
```

```
knn.pred=knn(train.X,test.X,train.Direction,k=5)
table(knn.pred,Direction.0910)
mean(knn.pred==Direction.0910)
```

12.

```
Power=function() {
  print(2^3)
}

Power2=function(x,a){
  print(x^a)
}

Power3=function(x,a) {
  result=x^a
  return(result)
}

PlotPower=function(x,a) {
  result=x^a
  plot(result,xlab="x",ylab="x^2",main="Power Function")
}

Power()

Power2(10,3)
Power2(8,17)
Power2(131,3)

Power3(1:10,2)
plot(Power3(1:10,2),log="y",xlab="x",ylab="x^2",main="Power Function")

PlotPower(1:10,3)
```

Chapter 5

5.

```
rm(list=ls())
set.seed(1)
library(ISLR)

# a) Fit logistic regression model
```

```
glm.fit = glm(default ~ income + balance, data = Default, family =
binomial)
glm.fit

# b) Estimate test error with validation approach
train = sample(1:nrow(Default), nrow(Default) / 2)
Default.test = Default[-train, ]
glm.fit = glm(default ~ income + balance, data = Default, family =
binomial,
              subset = train)
glm.pred = rep("No", nrow(Default) / 2)
glm.probs = predict(glm.fit, Default.test, type = "response")
glm.pred[glm.probs > 0.5] = "Yes"
mean(glm.pred != Default.test$default)

# d) Include student dummy variable
train = sample(1:nrow(Default), nrow(Default) / 2)
Default.test = Default[-train, ]
glm.fit = glm(default ~ income + balance + student, data = Default, family
= binomial,
              subset = train)
glm.pred = rep("No", nrow(Default) / 2)
glm.probs = predict(glm.fit, Default.test, type = "response")
glm.pred[glm.probs > 0.5] = "Yes"
mean(glm.pred != Default.test$default)
```

8.

```
rm(list=ls())

# a) Generate a simulated data set
set.seed(1)
x = rnorm(100)
y = x - 2*x^2 + rnorm(100)

# b) Scatterplot
plot(x, y)

# c) Find LOOCV errors from models
library(boot)
data.set = data.frame(x, y)
glm.fit = glm(y ~ x)
cv.err = cv.glm(data.set, glm.fit)
cv.err$delta
```

```
glm.fit = glm(y ~ poly(x, 2))
cv.err = cv.glm(data.set, glm.fit)
cv.err$delta

glm.fit = glm(y ~ poly(x, 3))
cv.err = cv.glm(data.set, glm.fit)
cv.err$delta

glm.fit = glm(y ~ poly(x, 4))
cv.err = cv.glm(data.set, glm.fit)
cv.err$delta

# d) Repeat with a different seed
set.seed(2)
glm.fit = glm(y ~ x)
cv.err = cv.glm(data.set, glm.fit)
cv.err$delta

glm.fit = glm(y ~ poly(x, 2))
cv.err = cv.glm(data.set, glm.fit)
cv.err$delta

glm.fit = glm(y ~ poly(x, 3))
cv.err = cv.glm(data.set, glm.fit)
cv.err$delta

glm.fit = glm(y ~ poly(x, 4))
cv.err = cv.glm(data.set, glm.fit)
cv.err$delta

summary(glm.fit)
```

Chapter 6

8.

```
rm(list=ls())
# a) Generate simulated data
set.seed(1)
X = rnorm(100)
eps = rnorm(100)
summary(X)
summary(eps)
```

```r
# b) Generate response vector Y
Y = 2 + 2*X + 0.5*X^2 + 4*X^3 + eps

# c) Perform best subset selection
library(leaps)
data.set = data.frame(x = X, y = Y)
regfit.full = regsubsets(y ~ poly(x, 10), data = data.set, nvmax = 10)
reg.summary = summary(regfit.full)

# Choose best models for C_p, BIC, and adjusted R^2
min.cp = which.min(reg.summary$cp)
min.bic = which.min(reg.summary$bic)
max.adjr2 = which.max(reg.summary$adjr2)

# Plot C_p, BIC, and adjusted R^2
plot(reg.summary$cp, type = "l", xlab = "# of Predictors", ylab = "Cp",
main = "Best Subset Selection")
points(x = min.cp, y = reg.summary$cp[min.cp], col = "red", pch = 16)

plot(reg.summary$bic, type = "l", xlab = "# of Predictors", ylab = "BIC",
main = "Best Subset Selection")
points(x = min.bic, y = reg.summary$bic[min.bic], col = "red", pch = 16)

plot(reg.summary$adjr2, type = "l", xlab = "# of Predictors", ylab =
"Adjusted R^2", main = "Best Subset Selection")
points(x = max.adjr2, y = reg.summary$adjr2[max.adjr2], col = "red", pch =
16)

coefficients(regfit.full, id = 3)

# d) Perform forward stepwise selection
regfit.fwd = regsubsets(y ~ poly(x, 10), data = data.set, nvmax = 10,
method = "forward")
reg.summary = summary(regfit.fwd)

# Choose best models for C_p, BIC, and adjusted R^2
min.cp = which.min(reg.summary$cp)
min.bic = which.min(reg.summary$bic)
max.adjr2 = which.max(reg.summary$adjr2)

# Plot C_p, BIC, and adjusted R^2
plot(reg.summary$cp, type = "l", xlab = "# of Predictors", ylab = "Cp",
main = "Forward Stepwise Selection")
points(x = min.cp, y = reg.summary$cp[min.cp], col = "red", pch = 16)
```

```r
plot(reg.summary$bic, type = "l", xlab = "# of Predictors", ylab = "BIC",
main = "Forward Stepwise Selection")
points(x = min.bic, y = reg.summary$bic[min.bic], col = "red", pch = 16)

plot(reg.summary$adjr2, type = "l", xlab = "# of Predictors", ylab =
"Adjusted R^2", main = "Forward Stepwise Selection")
points(x = max.adjr2, y = reg.summary$adjr2[max.adjr2], col = "red", pch =
16)

coefficients(regfit.fwd, id = 3)

# Perform backward stepwise selection
regfit.bwd = regsubsets(y ~ poly(x, 10), data = data.set, nvmax = 10,
method = "backward")
reg.summary = summary(regfit.bwd)

# Choose best models for C_p, BIC, and adjusted R^2
min.cp = which.min(reg.summary$cp)
min.bic = which.min(reg.summary$bic)
max.adjr2 = which.max(reg.summary$adjr2)

# Plot C_p, BIC, and adjusted R^2
plot(reg.summary$cp, type = "l", xlab = "# of Predictors", ylab = "Cp",
main = "Backward Stepwise Selection")
points(x = min.cp, y = reg.summary$cp[min.cp], col = "red", pch = 16)

plot(reg.summary$bic, type = "l", xlab = "# of Predictors", ylab = "BIC",
main = "Backward Stepwise Selection")
points(x = min.bic, y = reg.summary$bic[min.bic], col = "red", pch = 16)

plot(reg.summary$adjr2, type = "l", xlab = "# of Predictors", ylab =
"Adjusted R^2", main = "Backward Stepwise Selection")
points(x = max.adjr2, y = reg.summary$adjr2[max.adjr2], col = "red", pch =
16)

coefficients(regfit.bwd, id = 3)

# Fit a lasso model
library(glmnet)
x.mat =  model.matrix(y ~ poly(x, 10), data = data.set)[, -1]
lasso.mod = cv.glmnet(x.mat, Y, alpha = 1)
lasso.mod
best.lamda = lasso.mod$lambda.min
best.lamda
```

```r
plot(lasso.mod, main = "Lasso")

# Fit best model to full data set
best.model = glmnet(x.mat, Y, alpha = 1)
predict(best.model, s = best.lamda, type = "coefficients")

# f) Generate new response vector
Y = 2 + 7*X^7 + eps

# Best subset selection
data.set = data.frame(x = X, y = Y)
regfit.full = regsubsets(y ~ poly(x, 10), data = data.set, nvmax = 10)
reg.summary = summary(regfit.full)

# Choose best models for C_p, BIC, and adjusted R^2
which.min(reg.summary$cp)
which.min(reg.summary$bic)
which.max(reg.summary$adjr2)

coefficients(regfit.full, id = 1)
coefficients(regfit.full, id = 2)
coefficients(regfit.full, id = 4)

# Lasso
x.mat = model.matrix(y ~ poly(x, 10), data = data.set)[, -1]
lasso.mod = cv.glmnet(x.mat, Y, alpha = 1)
best.lambda = lasso.mod$lambda.min
best.lambda

# Fit best model to full data set
best.model = glmnet(x.mat, Y, alpha = 1)
predict(best.model, s = best.lamda, type = "coefficients")
```

9.

```r
rm(list=ls())
library(ISLR)
set.seed(1)

# a) Split data into training and test sets
len = nrow(College) / 2
train = sample(1:nrow(College), len)
test = -train
College.train = College[train,]
College.test = College[test,]
```

```
# b) Fit linear model and test
lm.fit = lm(Apps ~ ., data = College.train)
lm.pred = predict(lm.fit, College.test)
mean((College.test$Apps - lm.pred)^2)

# c) Fit a ridge regression model and test
library(glmnet)
train.mat = model.matrix(Apps ~ ., data = College.train)
test.mat = model.matrix(Apps ~ ., data = College.test)
ridge.mod = cv.glmnet(train.mat, College.train$Apps, alpha = 0)
best.lambda = ridge.mod$lambda.min
ridge.mod
ridge.pred = predict(ridge.mod, newx = test.mat, s = best.lambda)
mean((College.test$Apps - ridge.pred)^2)

# d) Fit a lasso model and test
lasso.mod = cv.glmnet(train.mat, College.train$Apps, alpha = 1)
best.lambda = lasso.mod$lambda.min
lasso.mod
lasso.pred = predict(lasso.mod, newx = test.mat, s = best.lambda)
mean((College.test$Apps - lasso.pred)^2)

# e) Fit a PCR model and test
library(pls)
pcr.fit = pcr(Apps ~ ., data = College.train, scale = T, validation =
"CV")
summary(pcr.fit)
validationplot(pcr.fit, val.type="MSEP", main = "PCR")
pcr.pred = predict(pcr.fit, College.test, ncomp = 10)
mean((College.test$Apps - pcr.pred)^2)

# f) Fit a PLS model and test
pls.fit = plsr(Apps ~ ., data = College.train, scale = T, validation =
"CV")
summary(pls.fit)
validationplot(pls.fit,val.type="MSEP", main = "PLS")
pls.pred = predict(pls.fit, College.test, ncomp=6)
mean((pls.pred - College.test$Apps)^2)
```

11.

```
rm(list=ls())
library(MASS)
set.seed(1)
```

```r
# Best subset selection
library(leaps)
predict.regsubsets = function(object, newdata, id, ...) {
  form = as.formula(object$call[[2]])
  mat = model.matrix(form, newdata)
  coefi = coef(object, id=id)
  xvars = names(coefi)
  mat[, xvars] %*% coefi
}

# Use 10-fold CV to choose best subset model with lowest RMSE
k = 10
p = ncol(Boston) - 1
folds = sample(rep(1:k), nrow(Boston), replace = T)
cv.errors = matrix(NA, k, p, dimnames = list(NULL, paste(1:p)))
for (i in 1:k) {
  best.fit = regsubsets(crim ~ ., data = Boston[folds != i,],
                        nvmax = p)
  for (j in 1:p) {
    pred = predict(best.fit, Boston[folds == i,], id = j)
    cv.errors[i, j] = mean((Boston$crim[folds == i] - pred)^2)
  }
}
mean.cv.errors = apply(cv.errors, 2, mean)
cv.rmse = sqrt(mean.cv.errors)
min.rmse = which.min(cv.rmse)
plot(cv.rmse, type = "b", xlab = "# of Predictors", ylab = "RMSE",
     main = "Best Subset Selection", pch = 16)
cv.rmse[min.rmse]


# Ridge regression
library(glmnet)
x = model.matrix(crim ~ ., Boston)[, -1]
y = Boston$crim
train = sample(1:nrow(Boston), nrow(Boston) / 2)
test = -train
cv.out = cv.glmnet(x[train,], y[train], alpha = 0)
plot(cv.out, main = "Ridge Regression")
best.lambda = cv.out$lambda.min
best.lambda
ridge.pred = predict(cv.out, s = best.lambda, newx = x[test,])
sqrt(mean((ridge.pred - y[test])^2)) # RMSE
```

```r
# Lasso
cv.out = cv.glmnet(x[train,], y[train], alpha = 1)
plot(cv.out, main = "Lasso")
best.lambda = cv.out$lambda.min
ridge.pred = predict(cv.out, s = best.lambda, newx = x[test,])
sqrt(mean((ridge.pred - y[test])^2)) # RMSE


# PCR
library(pls)
pcr.fit = pcr(crim ~ ., data = Boston, validation = "CV")
summary(pcr.fit)


# PLS
pls.fit = plsr(crim ~ ., data = Boston, validation = "CV")
summary(pls.fit)
```