Ans of the ques 1(a)

(a) See 3a.jpg.

(b)

all 5 lines >= 0

i. (squared) bias - decreases monotonically because increases in flexibility

yield a closer fit

ii. variance - increases monotonically because increases in flexibility yield

overfit

iii. training error - decreases monotonically because increases in flexibility

yield a closer fit

iv. test error - concave up curve because increase in flexibility yields a closer

fit before it overfits

v. Bayes (irreducible) error - defines the lower limit, the test error is bounded

below by the irreducible error due to variance in the error (epsilon) in the output

values (0 <= value). When the training error is lower than the irreducible error,

overfitting has taken place.

The Bayes error rate is defined for classification problems and is determined by

the ratio of data points which lie at the 'wrong' side of the decision boundary,

(0 <= value < 1).

Ans to ques no 2

The advantages for a very flexible approach for regression or classification

are obtaining a better fit for non-linear models, decreasing bias.

The disadvantages for a very flexible approach for regression or classification

are requires estimating a greater number of parameters, follow the noise too

closely (overfit), increasing variance.

A more flexible approach would be preferred to a less flexible approach when we

are interested in prediction and not the interpretability of the results.

A less flexible approach would be preferred to a more flexible approach when we

are interested in inference and the interpretability of the results.

Ans to ques no 3

(a)

college = read.csv("../data/College.csv")

# 8. (b)

fix(college)

rownames(college) = college[,1]

college = college[,-1]

fix(college)

# 8. (c)

# i.

summary(college)

# ii.

pairs(college[,1:10])

# iii.

plot(college$Private, college$Outstate)

# iv.

Elite = rep("No", nrow(college))

Elite[college$Top10perc>50] = "Yes"

Elite = as.factor(Elite)

college = data.frame(college, Elite)

summary(college$Elite)

plot(college$Elite, college$Outstate)

# v.

par(mfrow=c(2,2))

hist(college$Apps)

hist(college$perc.alumni, col=2)

hist(college$S.F.Ratio, col=3, breaks=10)

hist(college$Expend, breaks=100)

# vi.

par(mfrow=c(1,1))

plot(college$Outstate, college$Grad.Rate)

# High tuition correlates to high graduation rate.

plot(college$Accept / college$Apps, college$S.F.Ratio)

# Colleges with low acceptance rate tend to have low S:F ratio.

plot(college$Top10perc, college$Grad.Rate)

# Colleges with the most students from top 10% perc don't necessarily have

# the highest graduation rate. Also, rate > 100 is erroneous!

Ans of question no 4

Y = 50 + 20(gpa) + 0.07(iq) + 35(gender) + 0.01(gpa \* iq) - 10 (gpa \* gender)

(a) Y = 50 + 20 k\_1 + 0.07 k\_2 + 35 gender + 0.01(k\_1 \* k\_2) - 10 (k\_1 \* gender)

male: (gender = 0) 50 + 20 k\_1 + 0.07 k\_2 + 0.01(k\_1 \* k\_2)

female: (gender = 1) 50 + 20 k\_1 + 0.07 k\_2 + 35 + 0.01(k\_1 \* k\_2) - 10 (k\_1)

Once the GPA is high enough, males earn more on average. => iii.

(b) Y(Gender = 1, IQ = 110, GPA = 4.0)

= 50 + 20 \* 4 + 0.07 \* 110 + 35 + 0.01 (4 \* 110) - 10 \* 4

= 137.1

(c) False. We must examine the p-value of the regression coefficient to

determine if the interaction term is statstically significant or not.

Ans to ques no 5

5(a)

Auto = read.csv("../data/Auto.csv", header=T, na.strings="?")

Auto = na.omit(Auto)

summary(Auto)

**attach**(Auto)

lm.fit = lm(mpg ~ horsepower)

summary(lm.fit)

**i.**

Yes, there is a relationship between horsepower and mpg as deterined by testing the null hypothesis of all regression coefficients equal to zero. Since the F-statistic is far larger than 1 and the p-value of the F-statistic is close to zero we can reject the null hypothesis and state there is a statistically significant relationship between horsepower and mpg.

**ii.**

To calculate the residual error relative to the response we use the mean of the response and the RSE. The mean of mpg is 23.4459. The RSE of the lm.fit was 4.906 which indicates a percentage error of 20.9248%. The R2R2 of the lm.fit was about 0.6059, meaning 60.5948% of the variance in mpg is explained by horsepower.

**iii.**

The relationship between mpg and horsepower is negative. The more horsepower an automobile has the linear regression indicates the less mpg fuel efficiency the automobile will have.

**iv.**

predict(lm.fit, data.frame(horsepower=c(98)), interval="confidence")

predict(lm.fit, data.frame(horsepower=c(98)), interval="prediction")

## 8b.

plot(horsepower, mpg)

abline(lm.fit)

## 8c.

par(mfrow=c(2,2))

plot(lm.fit)

Ans to ques no 6

## 10a.

**library**(ISLR)

##

## Attaching package: 'ISLR'

##

## The following object is masked \_by\_ '.GlobalEnv':

##

## Auto

summary(Carseats)

## Sales CompPrice Income Advertising

## Min. : 0.00 Min. : 77 Min. : 21.0 Min. : 0.00

## 1st Qu.: 5.39 1st Qu.:115 1st Qu.: 42.8 1st Qu.: 0.00

## Median : 7.49 Median :125 Median : 69.0 Median : 5.00

## Mean : 7.50 Mean :125 Mean : 68.7 Mean : 6.63

## 3rd Qu.: 9.32 3rd Qu.:135 3rd Qu.: 91.0 3rd Qu.:12.00

## Max. :16.27 Max. :175 Max. :120.0 Max. :29.00

## Population Price ShelveLoc Age Education

## Min. : 10 Min. : 24 Bad : 96 Min. :25.0 Min. :10.0

## 1st Qu.:139 1st Qu.:100 Good : 85 1st Qu.:39.8 1st Qu.:12.0

## Median :272 Median :117 Medium:219 Median :54.5 Median :14.0

## Mean :265 Mean :116 Mean :53.3 Mean :13.9

## 3rd Qu.:398 3rd Qu.:131 3rd Qu.:66.0 3rd Qu.:16.0

## Max. :509 Max. :191 Max. :80.0 Max. :18.0

## Urban US

## No :118 No :142

## Yes:282 Yes:258

##

##

##

##

**attach**(Carseats)

lm.fit = lm(Sales~Price+Urban+US)

summary(lm.fit)

##

## Call:

## lm(formula = Sales ~ Price + Urban + US)

##

## Residuals:

## Min 1Q Median 3Q Max

## -6.921 -1.622 -0.056 1.579 7.058

##

## Coefficients:

## Estimate Std. Error t value Pr(>|t|)

## (Intercept) 13.04347 0.65101 20.04 < 2e-16 \*\*\*

## Price -0.05446 0.00524 -10.39 < 2e-16 \*\*\*

## UrbanYes -0.02192 0.27165 -0.08 0.94

## USYes 1.20057 0.25904 4.63 4.9e-06 \*\*\*

## ---

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

##

## Residual standard error: 2.47 on 396 degrees of freedom

## Multiple R-squared: 0.239, Adjusted R-squared: 0.234

## F-statistic: 41.5 on 3 and 396 DF, p-value: <2e-16

## 10b.

### Price

The linear regression suggests a relationship between price and sales given the low p-value of the t-statistic. The coefficient states a negative relationship between Price and Sales: as Price increases, Sales decreases.

### UrbanYes

The linear regression suggests that there isn’t a relationship between the location of the store and the number of sales based on the high p-value of the t-statistic.

### USYes

The linear regression suggests there is a relationship between whether the store is in the US or not and the amount of sales. The coefficient states a positive relationship between USYes and Sales: if the store is in the US, the sales will increase by approximately 1201 units.

## 10c.

Sales = 13.04 + -0.05 Price + -0.02 UrbanYes + 1.20 USYes

## 10d.

Price and USYes, based on the p-values, F-statistic, and p-value of the F-statistic.

## 10e.

lm.fit2 = lm(Sales ~ Price + US)

summary(lm.fit2)

##

## Call:

## lm(formula = Sales ~ Price + US)

##

## Residuals:

## Min 1Q Median 3Q Max

## -6.927 -1.629 -0.057 1.577 7.052

##

## Coefficients:

## Estimate Std. Error t value Pr(>|t|)

## (Intercept) 13.03079 0.63098 20.65 < 2e-16 \*\*\*

## Price -0.05448 0.00523 -10.42 < 2e-16 \*\*\*

## USYes 1.19964 0.25846 4.64 4.7e-06 \*\*\*

## ---

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

##

## Residual standard error: 2.47 on 397 degrees of freedom

## Multiple R-squared: 0.239, Adjusted R-squared: 0.235

## F-statistic: 62.4 on 2 and 397 DF, p-value: <2e-16

## 10f.

Based on the RSE and R^2 of the linear regressions, they both fit the data similarly, with linear regression from (e) fitting the data slightly better.

## 10g.

confint(lm.fit2)

## 2.5 % 97.5 %

## (Intercept) 11.79032 14.2713

## Price -0.06476 -0.0442

## USYes 0.69152 1.7078

## 10h.

plot(predict(lm.fit2), rstudent(lm.fit2))

 All studentized residuals appear to be bounded by -3 to 3, so not potential outliers are suggested from the linear regression.

par(mfrow=c(2,2))

plot(lm.fit2)

 There are a few observations that greatly exceed (p+1)/n(p+1)/n (0.0076) on the leverage-statistic plot that suggest that the corresponding points have high leverage.

Ans to ques no 7

### a

A general form of Ridge regression optimization looks like

Minimize: ∑i=1n(yi−β^0−∑j=1pβ^jxj)2+λ∑i=1pβ^2i∑i=1n(yi−β^0−∑j=1pβ^jxj)2+λ∑i=1pβ^i2

In this case, β^0=0β^0=0 and n=p=2n=p=2. So, the optimization looks like:

Minimize: (y1−β^1x11−β^2x12)2+(y2−β^1x21−β^2x22)2+λ(β^21+β^22)(y1−β^1x11−β^2x12)2+(y2−β^1x21−β^2x22)2+λ(β^12+β^22)

### b

Now we are given that, x11=x12=x1x11=x12=x1 and x21=x22=x2x21=x22=x2. We take derivatives of above expression with respect to both β1^β1^ and β2^β2^ and setting them equal to zero find that, β∗^1=x1y1+x2y2−β∗^2(x21+x22)λ+x21+x22β∗^1=x1y1+x2y2−β∗^2(x12+x22)λ+x12+x22 and β∗^2=x1y1+x2y2−β∗^1(x21+x22)λ+x21+x22β∗^2=x1y1+x2y2−β∗^1(x12+x22)λ+x12+x22

Symmetry in these expressions suggests that β∗^1=β∗^2β∗^1=β∗^2

### c

Like Ridge regression,

Minimize: (y1−β^1x11−β^2x12)2+(y2−β^1x21−β^2x22)2+λ(|β^1|+|β^2|)(y1−β^1x11−β^2x12)2+(y2−β^1x21−β^2x22)2+λ(|β^1|+|β^2|)

### d

Here is a geometric interpretation of the solutions for the equation in c above. We use the alternate form of Lasso constraints |β^1|+|β^2|<s|β^1|+|β^2|<s.

The Lasso constraint take the form |β^1|+|β^2|<s|β^1|+|β^2|<s, which when plotted take the familiar shape of a diamond centered at origin (0,0)(0,0). Next consider the squared optimization constraint (y1−β^1x11−β^2x12)2+(y2−β^1x21−β^2x22)2(y1−β^1x11−β^2x12)2+(y2−β^1x21−β^2x22)2. We use the facts x11=x12x11=x12, x21=x22x21=x22, x11+x21=0x11+x21=0, x12+x22=0x12+x22=0 and y1+y2=0y1+y2=0 to simplify it to

Minimize: 2.(y1−(β^1+β^2)x11)22.(y1−(β^1+β^2)x11)2.

This optimization problem has a simple solution: β^1+β^2=y1x11β^1+β^2=y1x11. This is a line parallel to the edge of Lasso-diamond β^1+β^2=sβ^1+β^2=s. Now solutions to the original Lasso optimization problem are contours of the function (y1−(β^1+β^2)x11)2(y1−(β^1+β^2)x11)2 that touch the Lasso-diamond β^1+β^2=sβ^1+β^2=s. Finally, as β^1β^1 and β^2β^2 very along the line β^1+β^2=y1x11β^1+β^2=y1x11, these contours touch the Lasso-diamond edge β^1+β^2=sβ^1+β^2=s at different points. As a result, the entire edge β^1+β^2=sβ^1+β^2=s is a potential solution to the Lasso optimization problem!

Similar argument can be made for the opposite Lasso-diamond edge: β^1+β^2=−sβ^1+β^2=−s.

Thus, the Lasso problem does not have a unique solution. The general form of solution is given by two line segments:

β^1+β^2=s;β^1≥0;β^2≥0β^1+β^2=s;β^1≥0;β^2≥0 and β^1+β^2=−s;β^1≤0;β^2≤0

**Ans to ques no 8**

**a**

Create 100 XX and ϵϵ variables

set.seed(1)

X = rnorm(100)

eps = rnorm(100)

**b**

We are selecting β0=3β0=3, β1=2β1=2, β2=−3β2=−3 and β3=0.3β3=0.3.

beta0 = 3

beta1 = 2

beta2 = -3

beta3 = 0.3

Y = beta0 + beta1 \* X + beta2 \* X^2 + beta3 \* X^3 + eps

**c**

Use regsubsetsregsubsets to select best model having polynomial of XX of degree 10

library(leaps)

data.full = data.frame(y = Y, x = X)

mod.full = regsubsets(y ~ poly(x, 10, raw = T), data = data.full, nvmax = 10)

mod.summary = summary(mod.full)

# Find the model size for best cp, BIC and adjr2

which.min(mod.summary$cp)

## [1] 3

which.min(mod.summary$bic)

## [1] 3

which.max(mod.summary$adjr2)

## [1] 3

# Plot cp, BIC and adjr2

plot(mod.summary$cp, xlab = "Subset Size", ylab = "Cp", pch = 20, type = "l")

points(3, mod.summary$cp[3], pch = 4, col = "red", lwd = 7)

plot(mod.summary$bic, xlab = "Subset Size", ylab = "BIC", pch = 20, type = "l")

points(3, mod.summary$bic[3], pch = 4, col = "red", lwd = 7)

plot(mod.summary$adjr2, xlab = "Subset Size", ylab = "Adjusted R2", pch = 20,

type = "l")

points(3, mod.summary$adjr2[3], pch = 4, col = "red", lwd = 7)

We find that with Cp, BIC and Adjusted R2 criteria, 33, 33, and 33 variable models are respectively picked.

coefficients(mod.full, id = 3)

## (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2

## 3.07627 2.35624 -3.16515

## poly(x, 10, raw = T)7

## 0.01047

All statistics pick X7X7 over X3X3. The remaining coefficients are quite close to ββ s.

**d**

We fit forward and backward stepwise models to the data.

mod.fwd = regsubsets(y ~ poly(x, 10, raw = T), data = data.full, nvmax = 10,

method = "forward")

mod.bwd = regsubsets(y ~ poly(x, 10, raw = T), data = data.full, nvmax = 10,

method = "backward")

fwd.summary = summary(mod.fwd)

bwd.summary = summary(mod.bwd)

which.min(fwd.summary$cp)

## [1] 3

which.min(bwd.summary$cp)

## [1] 3

which.min(fwd.summary$bic)

## [1] 3

which.min(bwd.summary$bic)

## [1] 3

which.max(fwd.summary$adjr2)

## [1] 3

which.max(bwd.summary$adjr2)

## [1] 4

# Plot the statistics

par(mfrow = c(3, 2))

plot(fwd.summary$cp, xlab = "Subset Size", ylab = "Forward Cp", pch = 20, type = "l")

points(3, fwd.summary$cp[3], pch = 4, col = "red", lwd = 7)

plot(bwd.summary$cp, xlab = "Subset Size", ylab = "Backward Cp", pch = 20, type = "l")

points(3, bwd.summary$cp[3], pch = 4, col = "red", lwd = 7)

plot(fwd.summary$bic, xlab = "Subset Size", ylab = "Forward BIC", pch = 20,

type = "l")

points(3, fwd.summary$bic[3], pch = 4, col = "red", lwd = 7)

plot(bwd.summary$bic, xlab = "Subset Size", ylab = "Backward BIC", pch = 20,

type = "l")

points(3, bwd.summary$bic[3], pch = 4, col = "red", lwd = 7)

plot(fwd.summary$adjr2, xlab = "Subset Size", ylab = "Forward Adjusted R2",

pch = 20, type = "l")

points(3, fwd.summary$adjr2[3], pch = 4, col = "red", lwd = 7)

plot(bwd.summary$adjr2, xlab = "Subset Size", ylab = "Backward Adjusted R2",

pch = 20, type = "l")

points(4, bwd.summary$adjr2[4], pch = 4, col = "red", lwd = 7)

We see that all statistics pick 33 variable models except backward stepwise with adjusted R2. Here are the coefficients

coefficients(mod.fwd, id = 3)

## (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2

## 3.07627 2.35624 -3.16515

## poly(x, 10, raw = T)7

## 0.01047

coefficients(mod.bwd, id = 3)

## (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2

## 3.07888 2.41982 -3.17724

## poly(x, 10, raw = T)9

## 0.00187

coefficients(mod.fwd, id = 4)

## (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2

## 3.112359 2.369859 -3.275727

## poly(x, 10, raw = T)4 poly(x, 10, raw = T)7

## 0.027674 0.009997

Here forward stepwise picks X7X7 over X3X3. Backward stepwise with 33 variables picks X9X9 while backward stepwise with 44 variables picks X4X4 and X7X7. All other coefficients are close to ββ s.

**e**

Training Lasso on the data

library(glmnet)

## Loading required package: Matrix

## Loading required package: lattice

## Loaded glmnet 1.9-5

xmat = model.matrix(y ~ poly(x, 10, raw = T), data = data.full)[, -1]

mod.lasso = cv.glmnet(xmat, Y, alpha = 1)

best.lambda = mod.lasso$lambda.min

best.lambda

## [1] 0.03991

plot(mod.lasso)

# Next fit the model on entire data using best lambda

best.model = glmnet(xmat, Y, alpha = 1)

predict(best.model, s = best.lambda, type = "coefficients")

## 11 x 1 sparse Matrix of class "dgCMatrix"

## 1

## (Intercept) 3.0398151

## poly(x, 10, raw = T)1 2.2303371

## poly(x, 10, raw = T)2 -3.1033193

## poly(x, 10, raw = T)3 .

## poly(x, 10, raw = T)4 .

## poly(x, 10, raw = T)5 0.0498411

## poly(x, 10, raw = T)6 .

## poly(x, 10, raw = T)7 0.0008068

## poly(x, 10, raw = T)8 .

## poly(x, 10, raw = T)9 .

## poly(x, 10, raw = T)10 .

Lasso also picks X5X5 over X3X3. It also picks X7X7 with negligible coefficient.

**f**

Create new Y with different β7=7β7=7.

beta7 = 7

Y = beta0 + beta7 \* X^7 + eps

# Predict using regsubsets

data.full = data.frame(y = Y, x = X)

mod.full = regsubsets(y ~ poly(x, 10, raw = T), data = data.full, nvmax = 10)

mod.summary = summary(mod.full)

# Find the model size for best cp, BIC and adjr2

which.min(mod.summary$cp)

## [1] 2

which.min(mod.summary$bic)

## [1] 1

which.max(mod.summary$adjr2)

## [1] 4

coefficients(mod.full, id = 1)

## (Intercept) poly(x, 10, raw = T)7

## 2.959 7.001

coefficients(mod.full, id = 2)

## (Intercept) poly(x, 10, raw = T)2 poly(x, 10, raw = T)7

## 3.0705 -0.1417 7.0016

coefficients(mod.full, id = 4)

## (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2

## 3.0763 0.2914 -0.1618

## poly(x, 10, raw = T)3 poly(x, 10, raw = T)7

## -0.2527 7.0091

We see that BIC picks the most accurate 1-variable model with matching coefficients. Other criteria pick additional variables.

xmat = model.matrix(y ~ poly(x, 10, raw = T), data = data.full)[, -1]

mod.lasso = cv.glmnet(xmat, Y, alpha = 1)

best.lambda = mod.lasso$lambda.min

best.lambda

## [1] 12.37

best.model = glmnet(xmat, Y, alpha = 1)

predict(best.model, s = best.lambda, type = "coefficients")

## 11 x 1 sparse Matrix of class "dgCMatrix"

## 1

## (Intercept) 3.820

## 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 6.797

## poly(x, 10, raw = T)8 .

## poly(x, 10, raw = T)9 .

## poly(x, 10, raw = T)10 .

Lasso also picks the best 1-variable model but intercet is quite off (3.83.8 vs 33).

**Ans to ques no 9**

### a

Load and split the College data

library(ISLR)

set.seed(11)

sum(is.na(College))

## [1] 0

train.size = dim(College)[1] / 2

train = sample(1:dim(College)[1], train.size)

test = -train

College.train = College[train, ]

College.test = College[test, ]

### b

NUmber of applications is the Apps variable.

lm.fit = lm(Apps~., data=College.train)

lm.pred = predict(lm.fit, College.test)

mean((College.test[, "Apps"] - lm.pred)^2)

## [1] 1538442

Test RSS is 15384421538442

### c

Pick λλ using College.train and report error on College.test

library(glmnet)

## Warning: package 'glmnet' was built under R version 2.15.2

## Loading required package: Matrix

## Warning: package 'Matrix' was built under R version 2.15.3

## Loading required package: lattice

## Warning: package 'lattice' was built under R version 2.15.3

## Loaded glmnet 1.9-3

train.mat = model.matrix(Apps~., data=College.train)

test.mat = model.matrix(Apps~., data=College.test)

grid = 10 ^ seq(4, -2, length=100)

mod.ridge = cv.glmnet(train.mat, College.train[, "Apps"], alpha=0, lambda=grid, thresh=1e-12)

lambda.best = mod.ridge$lambda.min

lambda.best

## [1] 18.74

ridge.pred = predict(mod.ridge, newx=test.mat, s=lambda.best)

mean((College.test[, "Apps"] - ridge.pred)^2)

## [1] 1608859

Test RSS is slightly higher that OLS, 16088591608859.

### d

Pick λλ using College.train and report error on College.test

mod.lasso = cv.glmnet(train.mat, College.train[, "Apps"], alpha=1, lambda=grid, thresh=1e-12)

lambda.best = mod.lasso$lambda.min

lambda.best

## [1] 21.54

lasso.pred = predict(mod.lasso, newx=test.mat, s=lambda.best)

mean((College.test[, "Apps"] - lasso.pred)^2)

## [1] 1635280

Again, Test RSS is slightly higher that OLS, 16352801635280.

The coefficients look like

mod.lasso = glmnet(model.matrix(Apps~., data=College), College[, "Apps"], alpha=1)

predict(mod.lasso, s=lambda.best, type="coefficients")

## 19 x 1 sparse Matrix of class "dgCMatrix"

## 1

## (Intercept) -6.038e+02

## (Intercept) .

## PrivateYes -4.235e+02

## Accept 1.455e+00

## Enroll -2.004e-01

## Top10perc 3.368e+01

## Top25perc -2.403e+00

## F.Undergrad .

## P.Undergrad 2.086e-02

## Outstate -5.782e-02

## Room.Board 1.246e-01

## Books .

## Personal 1.833e-05

## PhD -5.601e+00

## Terminal -3.314e+00

## S.F.Ratio 4.479e+00

## perc.alumni -9.797e-01

## Expend 6.968e-02

## Grad.Rate 5.160e+00

### e

Use validation to fit pcr

library(pls)

##

## Attaching package: 'pls'

##

## The following object(s) are masked from 'package:stats':

##

## loadings

pcr.fit = pcr(Apps~., data=College.train, scale=T, validation="CV")

validationplot(pcr.fit, val.type="MSEP")

pcr.pred = predict(pcr.fit, College.test, ncomp=10)

mean((College.test[, "Apps"] - data.frame(pcr.pred))^2)

## [1] 3014496

Test RSS for PCR is about 30144963014496.

### f

Use validation to fit pls

pls.fit = plsr(Apps~., data=College.train, scale=T, validation="CV")

validationplot(pls.fit, val.type="MSEP")

pls.pred = predict(pls.fit, College.test, ncomp=10)

mean((College.test[, "Apps"] - data.frame(pls.pred))^2)

## [1] 1508987

Test RSS for PLS is about 15089871508987.

### g

Results for OLS, Lasso, Ridge are comparable. Lasso reduces the F.UndergradF.Undergrad and BooksBooks variables to zero and shrinks coefficients of other variables. Here are the test R2R2 for all models.

test.avg = mean(College.test[, "Apps"])

lm.test.r2 = 1 - mean((College.test[, "Apps"] - lm.pred)^2) /mean((College.test[, "Apps"] - test.avg)^2)

ridge.test.r2 = 1 - mean((College.test[, "Apps"] - ridge.pred)^2) /mean((College.test[, "Apps"] - test.avg)^2)

lasso.test.r2 = 1 - mean((College.test[, "Apps"] - lasso.pred)^2) /mean((College.test[, "Apps"] - test.avg)^2)

pcr.test.r2 = 1 - mean((College.test[, "Apps"] - data.frame(pcr.pred))^2) /mean((College.test[, "Apps"] - test.avg)^2)

pls.test.r2 = 1 - mean((College.test[, "Apps"] - data.frame(pls.pred))^2) /mean((College.test[, "Apps"] - test.avg)^2)

barplot(c(lm.test.r2, ridge.test.r2, lasso.test.r2, pcr.test.r2, pls.test.r2), col="red", names.arg=c("OLS", "Ridge", "Lasso", "PCR", "PLS"), main="Test R-squared")

The plot shows that test R2R2 for all models except PCR are around 0.9, with PLS having slightly higher test R2R2 than others. PCR has a smaller test R2R2 of less than 0.8. All models except PCR predict college applications with high accuracy.

**Ans to ques no 10 (chapter 4,6)**

p(X)=exp(β0+β1X1+β2X2)1+exp(β0+β1X1+β2X2)X1=hoursstudied,X2=undergradGPAβ0=−6,β1=0.05,β2=1p(X)=exp⁡(β0+β1X1+β2X2)1+exp⁡(β0+β1X1+β2X2)X1=hoursstudied,X2=undergradGPAβ0=−6,β1=0.05,β2=1

**a.**

X=[40hours,3.5GPA]p(X)=exp(−6+0.05X1+X2)1+exp(−6+0.05X1+X2)=exp(−6+0.0540+3.5)1+exp(−6+0.0540+3.5)=exp(−0.5)1+exp(−0.5)=37.75%X=[40hours,3.5GPA]p(X)=exp⁡(−6+0.05X1+X2)1+exp⁡(−6+0.05X1+X2)=exp⁡(−6+0.0540+3.5)1+exp⁡(−6+0.0540+3.5)=exp⁡(−0.5)1+exp⁡(−0.5)=37.75%

**b.**

X=[X1hours,3.5GPA]p(X)=exp(−6+0.05X1+X2)1+exp(−6+0.05X1+X2)0.50=exp(−6+0.05X1+3.5)1+exp(−6+0.05X1+3.5)0.50(1+exp(−2.5+0.05X1))=exp(−2.5+0.05X1)0.50+0.50exp(−2.5+0.05X1))=exp(−2.5+0.05X1)0.50=0.50exp(−2.5+0.05X1)log(1)=−2.5+0.05X1X1=2.5/0.05=50hours

**Ans to ques no 11 (chapter 4,10)**

### a

library(ISLR)

summary(Weekly)

## Year Lag1 Lag2 Lag3

## Min. :1990 Min. :-18.195 Min. :-18.195 Min. :-18.195

## 1st Qu.:1995 1st Qu.: -1.154 1st Qu.: -1.154 1st Qu.: -1.158

## Median :2000 Median : 0.241 Median : 0.241 Median : 0.241

## Mean :2000 Mean : 0.151 Mean : 0.151 Mean : 0.147

## 3rd Qu.:2005 3rd Qu.: 1.405 3rd Qu.: 1.409 3rd Qu.: 1.409

## Max. :2010 Max. : 12.026 Max. : 12.026 Max. : 12.026

## Lag4 Lag5 Volume Today

## Min. :-18.195 Min. :-18.195 Min. :0.087 Min. :-18.195

## 1st Qu.: -1.158 1st Qu.: -1.166 1st Qu.:0.332 1st Qu.: -1.154

## Median : 0.238 Median : 0.234 Median :1.003 Median : 0.241

## Mean : 0.146 Mean : 0.140 Mean :1.575 Mean : 0.150

## 3rd Qu.: 1.409 3rd Qu.: 1.405 3rd Qu.:2.054 3rd Qu.: 1.405

## Max. : 12.026 Max. : 12.026 Max. :9.328 Max. : 12.026

## Direction

## Down:484

## Up :605

##

##

##

##

pairs(Weekly)

cor(Weekly[, -9])

## Year Lag1 Lag2 Lag3 Lag4 Lag5 Volume

## Year 1.00000 -0.032289 -0.03339 -0.03001 -0.031128 -0.030519 0.84194

## Lag1 -0.03229 1.000000 -0.07485 0.05864 -0.071274 -0.008183 -0.06495

## Lag2 -0.03339 -0.074853 1.00000 -0.07572 0.058382 -0.072499 -0.08551

## Lag3 -0.03001 0.058636 -0.07572 1.00000 -0.075396 0.060657 -0.06929

## Lag4 -0.03113 -0.071274 0.05838 -0.07540 1.000000 -0.075675 -0.06107

## Lag5 -0.03052 -0.008183 -0.07250 0.06066 -0.075675 1.000000 -0.05852

## Volume 0.84194 -0.064951 -0.08551 -0.06929 -0.061075 -0.058517 1.00000

## Today -0.03246 -0.075032 0.05917 -0.07124 -0.007826 0.011013 -0.03308

## Today

## Year -0.032460

## Lag1 -0.075032

## Lag2 0.059167

## Lag3 -0.071244

## Lag4 -0.007826

## Lag5 0.011013

## Volume -0.033078

## Today 1.000000

Year and Volume appear to have a relationship. No other patterns are discernible.

### b

attach(Weekly)

glm.fit = glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume, data = Weekly,

family = binomial)

summary(glm.fit)

##

## Call:

## glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +

## Volume, family = binomial, data = Weekly)

##

## Deviance Residuals:

## Min 1Q Median 3Q Max

## -1.695 -1.256 0.991 1.085 1.458

##

## Coefficients:

## Estimate Std. Error z value Pr(>|z|)

## (Intercept) 0.2669 0.0859 3.11 0.0019 \*\*

## Lag1 -0.0413 0.0264 -1.56 0.1181

## Lag2 0.0584 0.0269 2.18 0.0296 \*

## Lag3 -0.0161 0.0267 -0.60 0.5469

## Lag4 -0.0278 0.0265 -1.05 0.2937

## Lag5 -0.0145 0.0264 -0.55 0.5833

## Volume -0.0227 0.0369 -0.62 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

##

## Number of Fisher Scoring iterations: 4

Lag 2 appears to have some statistical significance with a Pr(>|z|) = 3%.

### c

glm.probs = predict(glm.fit, type = "response")

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

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

table(glm.pred, Direction)

## Direction

## glm.pred Down Up

## Down 54 48

## Up 430 557

Percentage of currect predictions: (54+557)/(54+557+48+430) = 56.1%. Weeks the market goes up the logistic regression is right most of the time, 557/(557+48) = 92.1%. Weeks the market goes up the logistic regression is wrong most of the time 54/(430+54) = 11.2%.

### d

train = (Year < 2009)

Weekly.0910 = Weekly[!train, ]

glm.fit = glm(Direction ~ Lag2, data = Weekly, family = binomial, subset = train)

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

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

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

Direction.0910 = Direction[!train]

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

### e

library(MASS)

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

lda.pred = predict(lda.fit, 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

## f

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

qda.class = predict(qda.fit, Weekly.0910)$class

table(qda.class, Direction.0910)

## Direction.0910

## qda.class Down Up

## Down 0 0

## Up 43 61

mean(qda.class == Direction.0910)

## [1] 0.5865

A correctness of 58.7% even though it picked Up the whole time!

### g

library(class)

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

### h

Logistic regression and LDA methods provide similar test error rates.

### i

# Logistic regression with Lag2:Lag1

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

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

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

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

Direction.0910 = Direction[!train]

table(glm.pred, Direction.0910)

## Direction.0910

## glm.pred Down Up

## Down 1 1

## Up 42 60

mean(glm.pred == Direction.0910)

## [1] 0.5865

# LDA with Lag2 interaction with Lag1

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

lda.pred = predict(lda.fit, Weekly.0910)

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

## [1] 0.5769

# QDA with sqrt(abs(Lag2))

qda.fit = qda(Direction ~ Lag2 + sqrt(abs(Lag2)), data = Weekly, subset = train)

qda.class = predict(qda.fit, Weekly.0910)$class

table(qda.class, Direction.0910)

## Direction.0910

## qda.class Down Up

## Down 12 13

## Up 31 48

mean(qda.class == Direction.0910)

## [1] 0.5769

# KNN k =10

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

table(knn.pred, Direction.0910)

## Direction.0910

## knn.pred Down Up

## Down 17 18

## Up 26 43

mean(knn.pred == Direction.0910)

## [1] 0.5769

# KNN k = 100

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

table(knn.pred, Direction.0910)

## Direction.0910

## knn.pred Down Up

## Down 9 12

## Up 34 49

mean(knn.pred == Direction.0910)

## [1] 0.5577

Out of these permutations, the original LDA and logistic regression have better performance in terms of test error rate.

**Ans to ques no 12**

### a

library(ISLR)

summary(Auto)

## mpg cylinders displacement horsepower

## Min. : 9.0 Min. :3.00 Min. : 68 Min. : 46.0

## 1st Qu.:17.0 1st Qu.:4.00 1st Qu.:105 1st Qu.: 75.0

## Median :22.8 Median :4.00 Median :151 Median : 93.5

## Mean :23.4 Mean :5.47 Mean :194 Mean :104.5

## 3rd Qu.:29.0 3rd Qu.:8.00 3rd Qu.:276 3rd Qu.:126.0

## Max. :46.6 Max. :8.00 Max. :455 Max. :230.0

##

## weight acceleration year origin

## Min. :1613 Min. : 8.0 Min. :70 Min. :1.00

## 1st Qu.:2225 1st Qu.:13.8 1st Qu.:73 1st Qu.:1.00

## Median :2804 Median :15.5 Median :76 Median :1.00

## Mean :2978 Mean :15.5 Mean :76 Mean :1.58

## 3rd Qu.:3615 3rd Qu.:17.0 3rd Qu.:79 3rd Qu.:2.00

## Max. :5140 Max. :24.8 Max. :82 Max. :3.00

##

## name

## amc matador : 5

## ford pinto : 5

## toyota corolla : 5

## amc gremlin : 4

## amc hornet : 4

## chevrolet chevette: 4

## (Other) :365

attach(Auto)

mpg01 = rep(0, length(mpg))

mpg01[mpg > median(mpg)] = 1

Auto = data.frame(Auto, mpg01)

### b

cor(Auto[, -9])

## mpg cylinders displacement horsepower weight

## mpg 1.0000 -0.7776 -0.8051 -0.7784 -0.8322

## cylinders -0.7776 1.0000 0.9508 0.8430 0.8975

## displacement -0.8051 0.9508 1.0000 0.8973 0.9330

## horsepower -0.7784 0.8430 0.8973 1.0000 0.8645

## weight -0.8322 0.8975 0.9330 0.8645 1.0000

## acceleration 0.4233 -0.5047 -0.5438 -0.6892 -0.4168

## year 0.5805 -0.3456 -0.3699 -0.4164 -0.3091

## origin 0.5652 -0.5689 -0.6145 -0.4552 -0.5850

## mpg01 0.8369 -0.7592 -0.7535 -0.6671 -0.7578

## acceleration year origin mpg01

## mpg 0.4233 0.5805 0.5652 0.8369

## cylinders -0.5047 -0.3456 -0.5689 -0.7592

## displacement -0.5438 -0.3699 -0.6145 -0.7535

## horsepower -0.6892 -0.4164 -0.4552 -0.6671

## weight -0.4168 -0.3091 -0.5850 -0.7578

## acceleration 1.0000 0.2903 0.2127 0.3468

## year 0.2903 1.0000 0.1815 0.4299

## origin 0.2127 0.1815 1.0000 0.5137

## mpg01 0.3468 0.4299 0.5137 1.0000

pairs(Auto) # doesn't work well since mpg01 is 0 or 1

Anti-correlated with cylinders, weight, displacement, horsepower. (mpg, of course)

### c

train = (year%%2 == 0) # if the year is even

test = !train

Auto.train = Auto[train, ]

Auto.test = Auto[test, ]

mpg01.test = mpg01[test]

### d

# LDA

library(MASS)

lda.fit = lda(mpg01 ~ cylinders + weight + displacement + horsepower, data = Auto,

subset = train)

lda.pred = predict(lda.fit, Auto.test)

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

## [1] 0.1264

12.6% test error rate.

### e

# QDA

qda.fit = qda(mpg01 ~ cylinders + weight + displacement + horsepower, data = Auto,

subset = train)

qda.pred = predict(qda.fit, Auto.test)

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

## [1] 0.1319

13.2% test error rate.

### f

# Logistic regression

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

family = binomial, subset = train)

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

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

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

mean(glm.pred != mpg01.test)

## [1] 0.1209

12.1% test error rate.

### 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(k=1)

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

mean(knn.pred != mpg01.test)

## [1] 0.1538

# KNN(k=10)

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

mean(knn.pred != mpg01.test)

## [1] 0.1648

# KNN(k=100)

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

mean(knn.pred != mpg01.test)

## [1] 0.1429

k=1, 15.4% test error rate. k=10, 16.5% test error rate. k=100, 14.3% test error rate. K of 100 seems to perform the best. 100 nearest neighbors.