Stats202 Homework 3

By Fang Lin (Stanford ID # 06166564)

July 27, 2016

Problem 1 (p.260, ex3)

- (a)

  Answer: Will steadily decreases. Since s increases from 0, traning error for s=0 is the maximum and it keep decreasing to the Ordinary Least Square RSS.
- (b)

  Answer: Decrease initially, and then eventually start increasing in a U shape. When s=0, all beta=0, so the model is under-fitting. As increasing s, beta becomes more fitting on the the test data, while after one point it will become over fitting.
- (c)

  Answer: Steadily increase. Variance will keep increasing as long as the s increasing from 0.
- (d)

  Answer: Steadily decrease. Bias will keep decreasing as long as the beta inceasing from 0. When s=0, the model predict a constant so the bias is highest.
- (e)

  Answer: Remain constant. Irreducible error is model independent.

Problem 2 (p.260, ex4)

- (a)
  Answer: Steadily increase. As increasing numda from 0, beta decreases from least square estimate values to 0.
- (b)
  Answer: Decrease initially, and then eventually start increasing in a U shape. When numda=0, all beta have their least square estimate values. So, in this case, the model can fit the tranining data best, but overfit cause high RSS. As numbda increasing, beta start decreasing to 0, so overfitting reduced, then as beta approach to 0, the model becomes too simple while test RSS will increase.
- (c)
  Answer: Steadily decrease. Variance will keep decreasing as long as the numbda increasing from 0.
- (d)
  Answer: Steadily increase. Bias will keep increasing as long as the numbda inceasing from 0. When numbda=0, the model has the least bias.
- (e)
  Answer: Remain constant. Irreducible error is model independent.

Problem 3 (p.262, ex8)

- (a)

  Answer:

```
> set.seed(1)
> X = rnorm(100)
> eps = rnorm(100)
```

- (b)

  Answer:

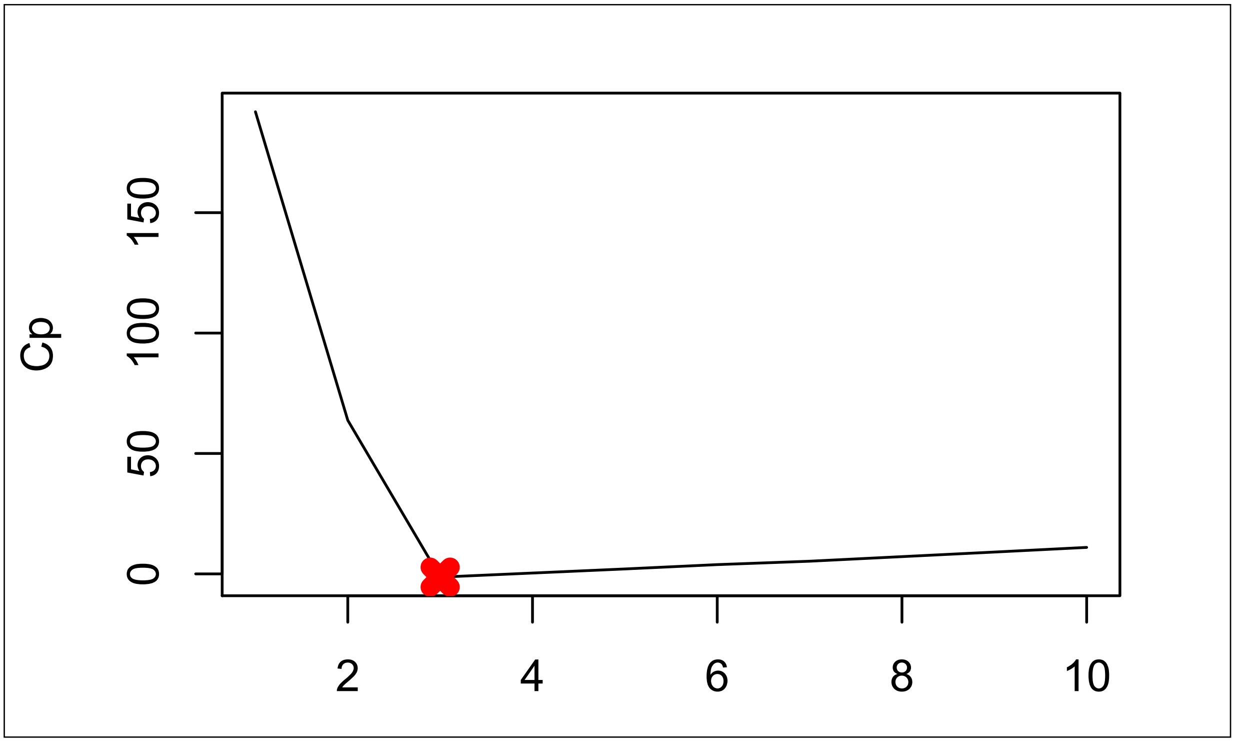
  We select beta0=1, beta1=2, beta2=-1, beta3=0.5.

```
> beta0=1
> beta1=2
> beta2=-1
> beta3=0.5
> Y = beta0 + beta1 * X + beta2 * X^2 + beta3 * X^3 + eps
```

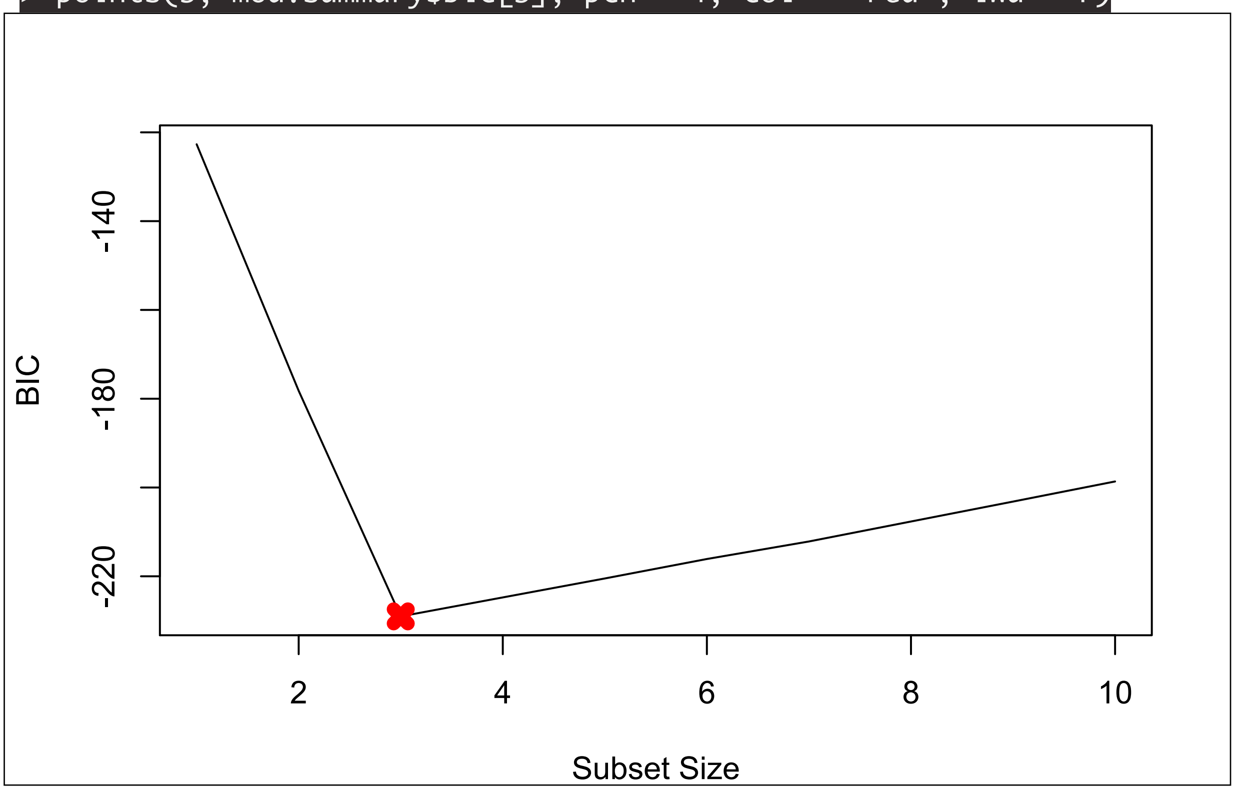
- (c)

  Answer:

```
> install.packages("leaps")
> 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)
> which.min(mod.summary$cp)
[1] 3
> which.min(mod.summary$bic)
[1] 3
> which.max(mod.summary$adjr2)
[1] 3
> plot(mod.summary$cp, xlab = "Subset Size", ylab = "Cp", pch = 20,
type = "l")
> points(3, mod.summary$cp[3], pch = 4, col = "red", lwd = 7)
```

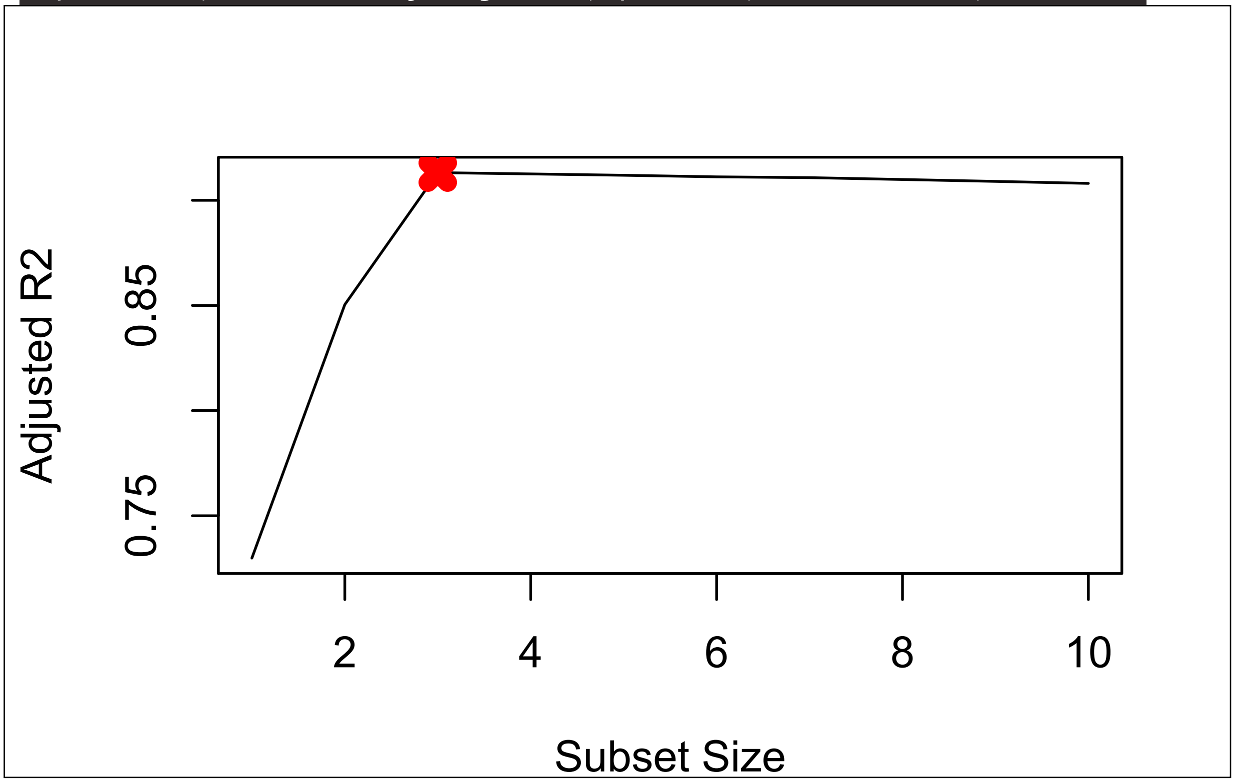



```
> plot(mod.summary$bic, xlab = "Subset Size", ylab = "BIC", pch = 20,
type = "l")
> points(3, mod.summary$bic[3], pch = 4, col = "red", lwd = 7)
```




```
> 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)
```



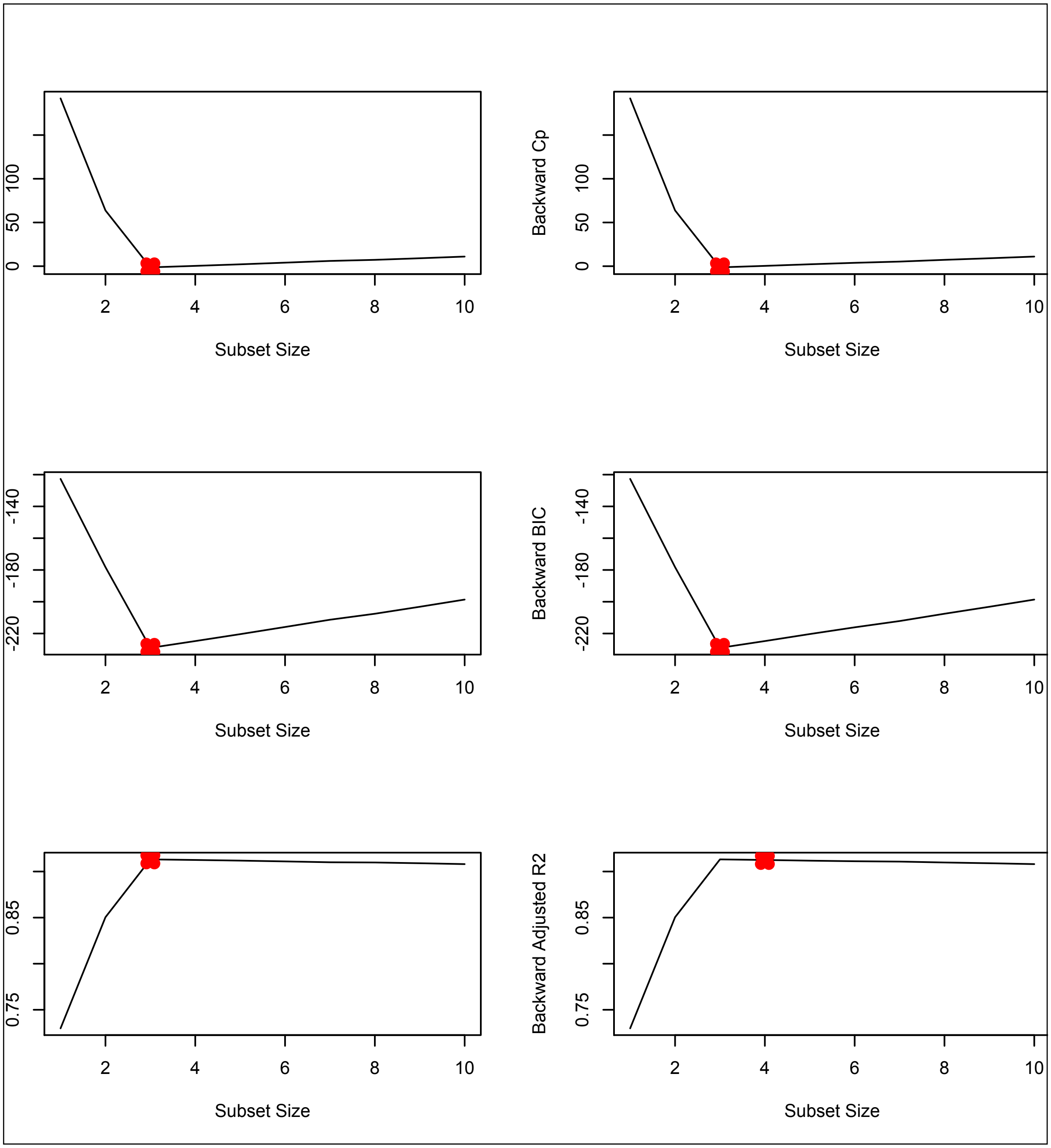

```
> plot(mod.summary$adjr2, xlab = "Subset Size", ylab = "Adjusted R2",
pch = 20,
+     type = "l")
> points(3, mod.summary$adjr2[3], pch = 4, col = "red", lwd = 7)
> coefficients(mod.full, id = 3)
          (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2
           1.07219472            2.44514720           -1.15676236
poly(x, 10, raw = T)5
           0.09022577
```

Answer: With Cp, BIC and Adjusted R2 criteria, 3, 3, 3 variable models are picked. All statistics pick X^5 over X^3. The remaining coefficients are quite close to betas.

- (d)

```
> 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$adjr2)
[1] 3
> par(mfrow = c(3, 2))
> plot(fwd.summary$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)
```




```
> coefficients(mod.fwd, id = 3)
          (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2
           1.07219472            2.44514720           -1.15676236
poly(x, 10, raw = T)5
           0.09022577
> coefficients(mod.bwd, id = 3)
          (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2
           1.07219472            2.44514720           -1.15676236
poly(x, 10, raw = T)5
```

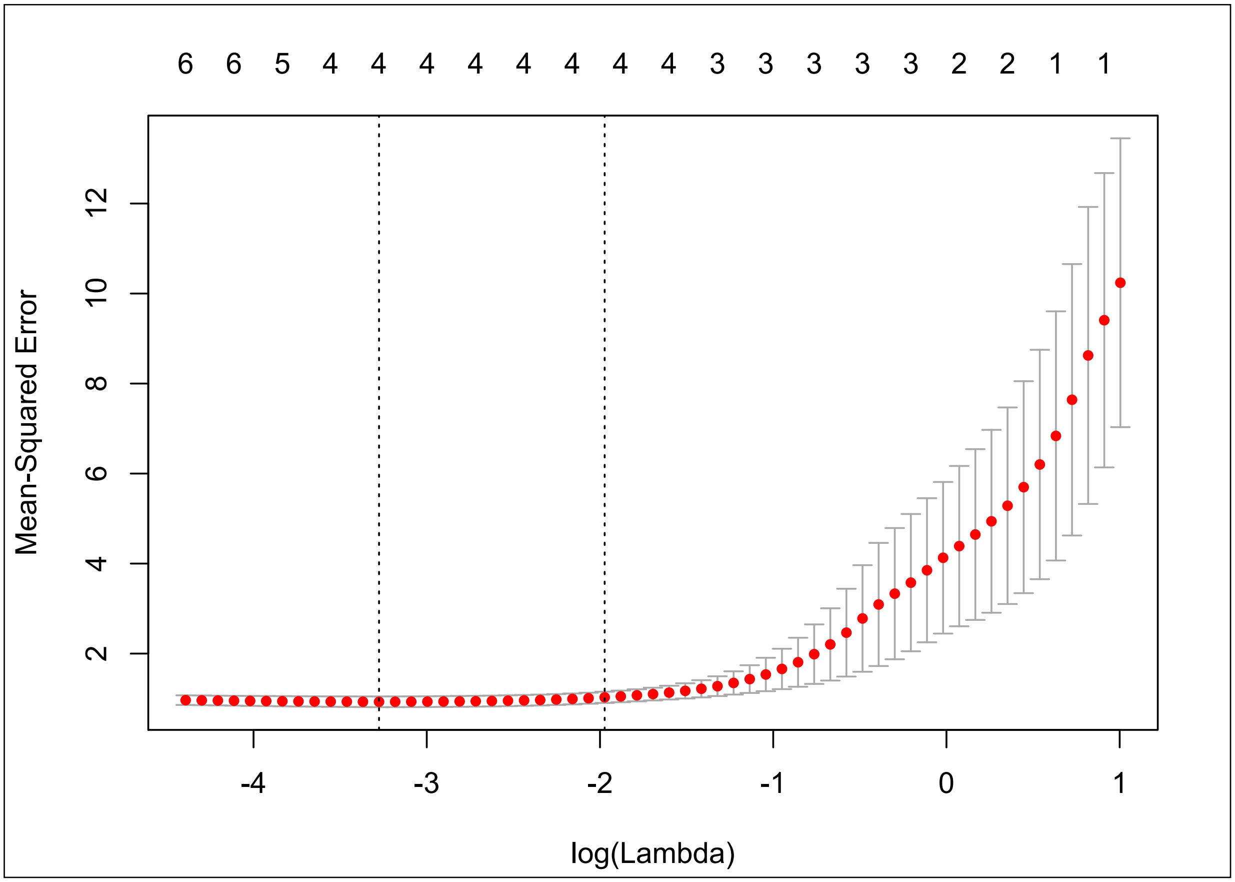
```
           0.09022577
> coefficients(mod.fwd, id = 4)
          (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2
           1.11309290            2.46288862           -1.28112227
poly(x, 10, raw = T)4 poly(x, 10, raw = T)5
           0.03074107            0.08769746
```

Answer: Forware stepwise picks X^5 over X^3. Backward stepwise with 3 variable picks X^5 while backward stepwise with 4 variables picks X^4 abd X^5. All other coefficients are close to betas.

- (e)

```
> library(glmnet)
Loading required package: Matrix
Loading required package: foreach
foreach: simple, scalable parallel programming from Revolution
Analytics
Use Revolution R for scalability, fault tolerance and more.
http://www.revolutionanalytics.com
Loaded glmnet 2.0-5
```

```
> 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] 0.03779912
> plot(mod.lasso)
```




```
> 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)          1.04103683
poly(x, 10, raw = T)1   2.28787832
poly(x, 10, raw = T)2  -1.10681293
poly(x, 10, raw = T)3   0.13650159
poly(x, 10, raw = T)4   .
poly(x, 10, raw = T)5   0.06464279
poly(x, 10, raw = T)6   .
poly(x, 10, raw = T)7   .
poly(x, 10, raw = T)8   .
poly(x, 10, raw = T)9   .
poly(x, 10, raw = T)10  .
```

Answer: Lesso picks X^3 also little bit X^5.

- (e)

```
> 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$adjr2)
[1] 4
> coefficients(mod.full, id = 1)
          (Intercept) poly(x, 10, raw = T)7
            0.9589402             7.0007705
> coefficients(mod.full, id = 2)
          (Intercept) poly(x, 10, raw = T)2 poly(x, 10, raw = T)7
            1.0704904            -0.1417084             7.0015552
> coefficients(mod.full, id = 4)
          (Intercept) poly(x, 10, raw = T)1 poly(x, 10, raw = T)2
            1.0762524             0.2914016            -0.1617671
poly(x, 10, raw = T)3 poly(x, 10, raw = T)7
           -0.2526527             7.0091338
> xmat = model.matrix(y ~ poly(x, 10, raw = T), data = data.full)[,
-1]
> mod.lasso = cv.glmnet(xmat, Y, alpha = 1)
> best.lambda = mod.lasso$lambda.min
> best.lambda
[1] 13.57478
> 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)           1.904188
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 6.776797
poly(x, 10, raw = T)8 .
```

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

Answer: We see BIC and Lesso both pick the best 1-variable.

Problem 4 (p.263, ex9)

- (a)

```
> 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)

```
> lm.fit = lm(Apps~., data=College.train)
> lm.pred = predict(lm.fit, College.test)
> mean((College.test[, "Apps"] - lm.pred)^2)
[1] 1538442
```

- (c)

```
> library(glmnet)
> 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] 18.73817
> ridge.pred = predict(mod.ridge, newx=test.mat, s=lambda.best)
> mean((College.test[, "Apps"] - ridge.pred)^2)
[1] 1608859
```

Answer: The RSS is slightly higher that OLS, 1608859.

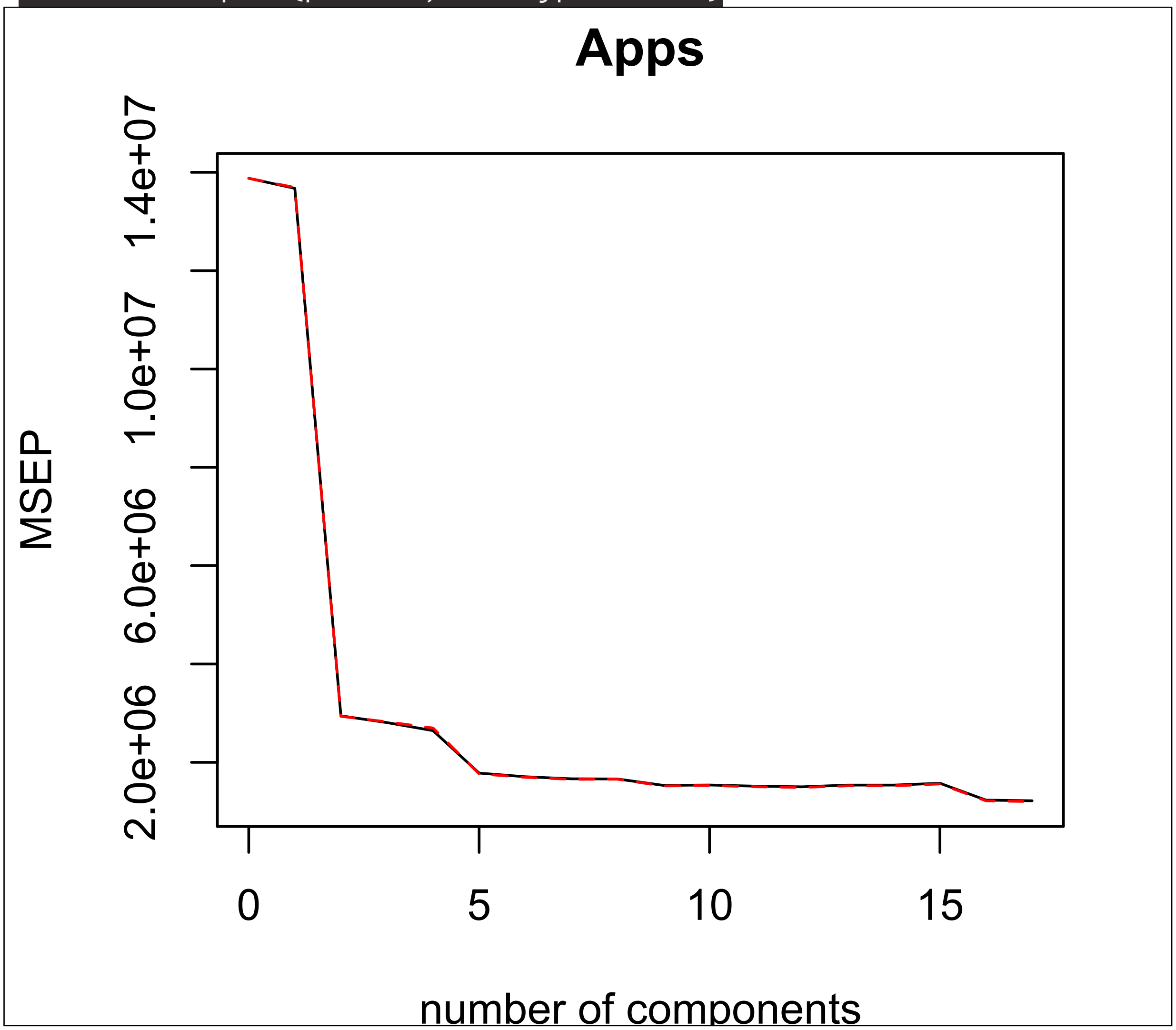
- (d)

```
> 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] 21.54435
> lasso.pred = predict(mod.lasso, newx=test.mat, s=lambda.best)
> mean((College.test[, "Apps"] - lasso.pred)^2)
[1] 1635280
```

- (e)

```
> library(pls)
```

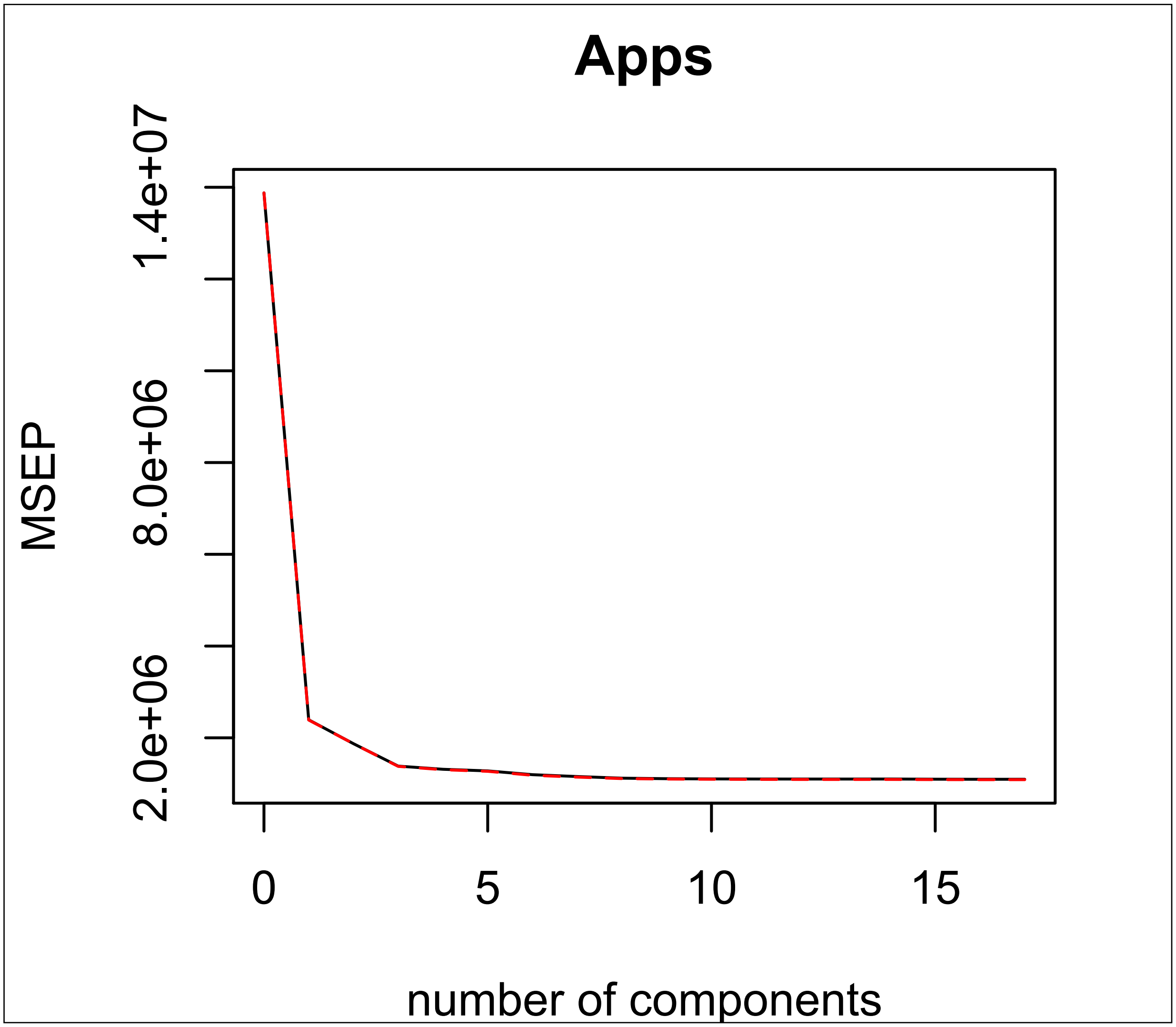
```
Attaching package: 'pls'
The following object is masked from 'package:stats':
    loadings
> pcr.fit = pcr(Apps~., data=College.train, scale=T, validation="CV")
> validationplot(pcr.fit, val.type="MSEP")
```




```
> pcr.pred = predict(pcr.fit, College.test, ncomp=10)
> mean((College.test[, "Apps"] - data.frame(pcr.pred))^2)
[1] 3014496
```

- (f)

```
> pls.fit = plsr(Apps~., data=College.train, scale=T,
validation="CV")
> validationplot(pls.fit, val.type="MSEP")
```

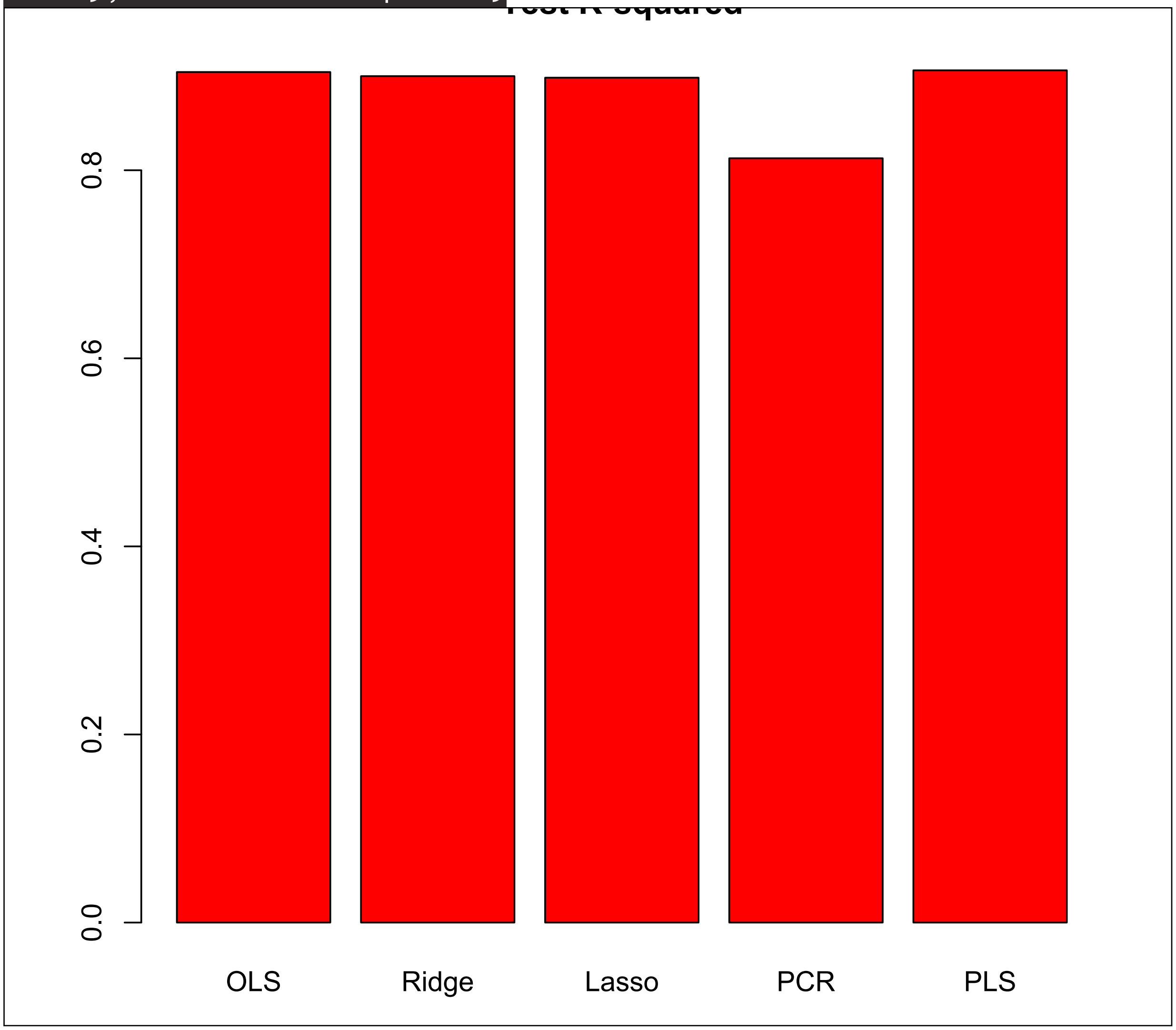



```
> pls.pred = predict(pls.fit, College.test, ncomp=10)
> mean((College.test[, "Apps"] - data.frame(pls.pred))^2)
[1] 1508987
```

- (g)

```
> 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")
```

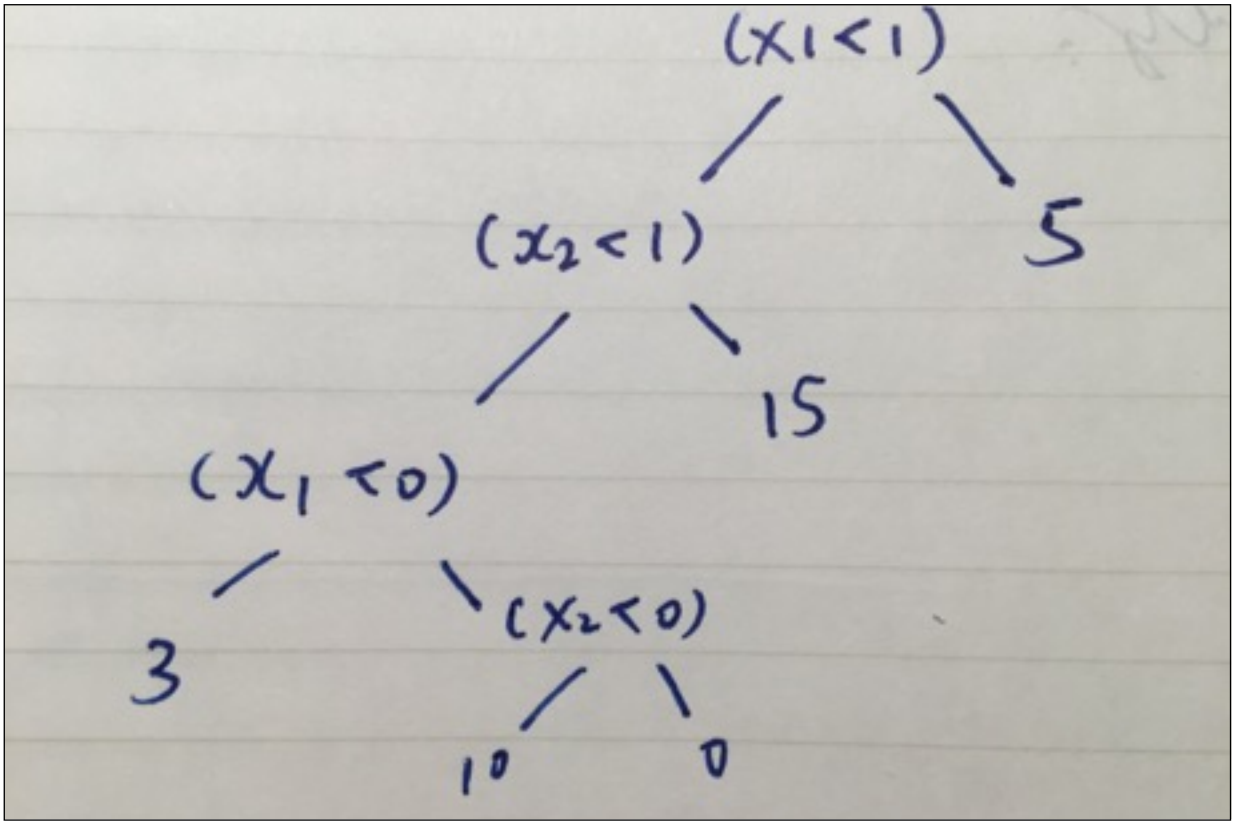


Answer: The plot shows test R^2 for different models other than PCR are near 0.9. PCR has a smaller test square root. All models except PCR predict college applications with hight accuracy.

Problem 5 (p.332, ex4)

(a)

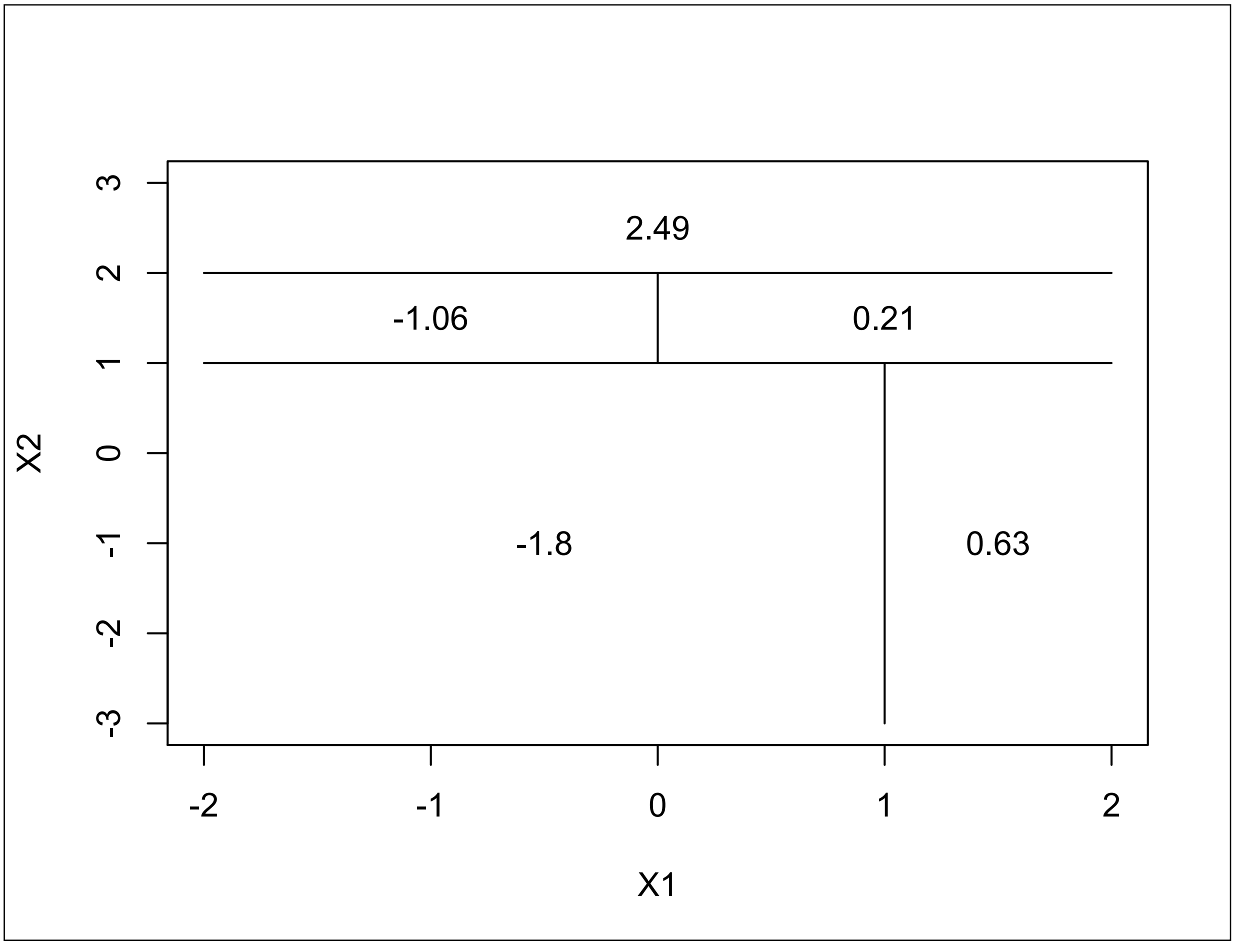
Answer:



(b)

```
> par(xpd = NA)
> plot(NA, NA, type = "n", xlim = c(-2, 2), ylim = c(-3, 3), xlab =
"X1", ylab = "X2")
> # X2 < 1
> lines(x = c(-2, 2), y = c(1, 1))
> # X1 < 1 with X2 < 1
> lines(x = c(1, 1), y = c(-3, 1))
> text(x = (-2 + 1)/2, y = -1, labels = c(-1.8))
> text(x = 1.5, y = -1, labels = c(0.63))
> # X2 < 2 with X2 >= 1
> lines(x = c(-2, 2), y = c(2, 2))
> text(x = 0, y = 2.5, labels = c(2.49))
> # X1 < 0 with X2<2 and X2>=1
> lines(x = c(0, 0), y = c(1, 2))
> text(x = -1, y = 1.5, labels = c(-1.06))
> text(x = 1, y = 1.5, labels = c(0.21))
```

Answer:



Problem 6 (p.332, ex5)

(a)

```
> p = c(0.1, 0.15, 0.2, 0.2, 0.55, 0.6, 0.6, 0.65, 0.7, 0.75)
> sum(p >= 0.5) > sum(p < 0.5)
[1] TRUE
> mean(p)
[1] 0.45
```

Answer:

Using 50% threshold, the number of red predictions is larger than the number of green predictions. Average approach will choose green, since the average of the probabilities is lesst than the 50 threshold.

Problem 7 (p.333, ex8)

(a)

Answer:

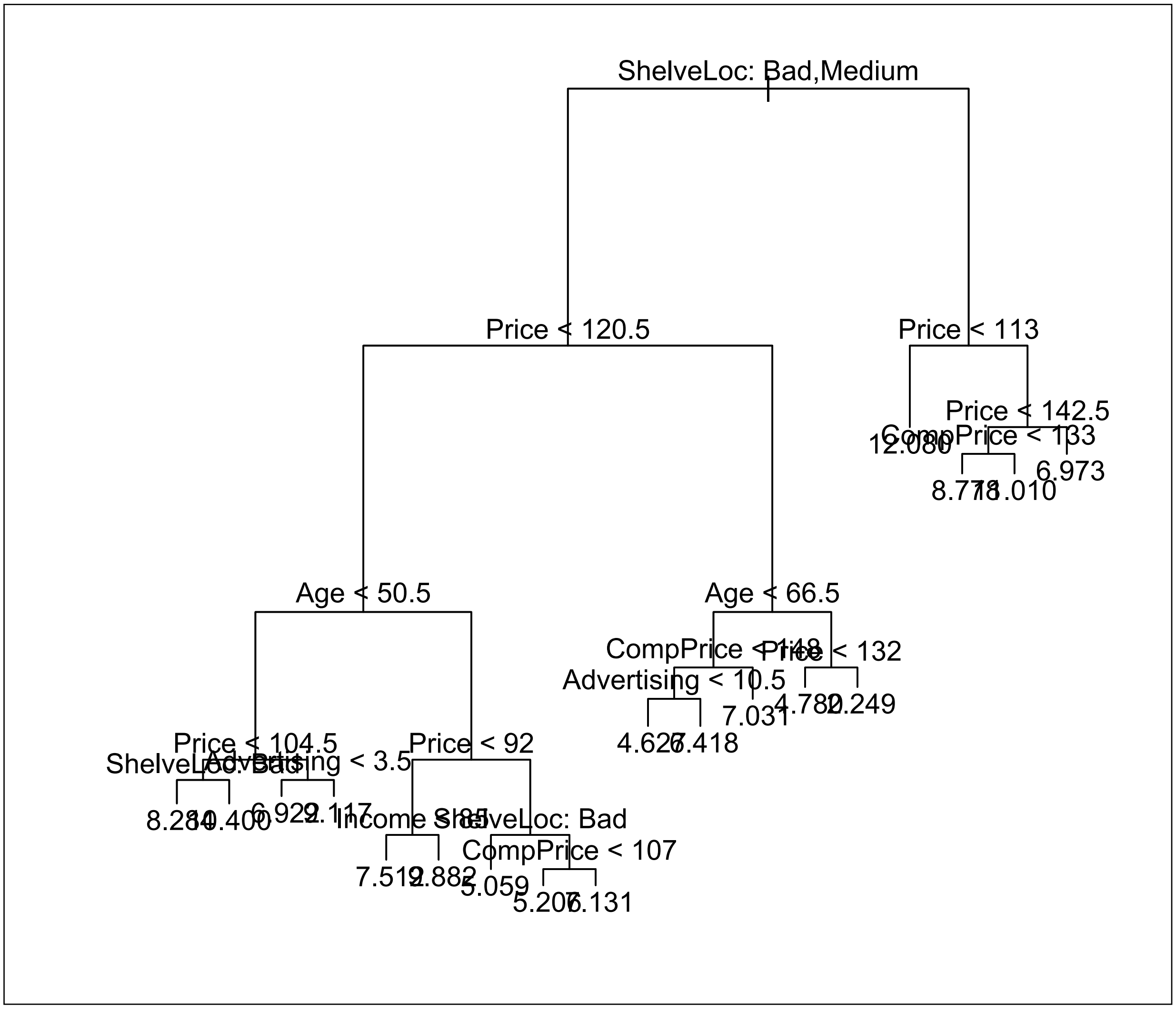
```
> library(ISLR)
> attach(Carseats)
> set.seed(1)
>
> train = sample(dim(Carseats)[1], dim(Carseats)[1]/2)
> Carseats.train = Carseats[train, ]
> Carseats.test = Carseats[-train, ]
```

(b)

```
> library(tree)
> tree.carseats = tree(Sales ~ ., data = Carseats.train)
> summary(tree.carseats)

Regression tree:
tree(formula = Sales ~ ., data = Carseats.train)
Variables actually used in tree construction:
[1] "ShelveLoc"   "Price"       "Age"         "Advertising" "Income"
[6] "CompPrice"
Number of terminal nodes:  18
Residual mean deviance:  2.36 = 429.5 / 182
Distribution of residuals:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-4.2570 -1.0360  0.1024  0.0000  0.9301  3.9130
> plot(tree.carseats)
> text(tree.carseats, pretty = 0)
```

Answer:




```
> pred.carseats = predict(tree.carseats, Carseats.test)
> mean((Carseats.test$Sales - pred.carseats)^2)
[1] 4.148897
```

(c)

```
> cv.carseats = cv.tree(tree.carseats, FUN = prune.tree)
> par(mfrow = c(1, 2))
> # Best size = 9
> pruned.carseats = prune.tree(tree.carseats, best = 9)
> par(mfrow = c(1, 1))
> plot(pruned.carseats)
> text(pruned.carseats, pretty = 0):
```

Answer:

```
> pred.pruned = predict(pruned.carseats, Carseats.test)
> mean((Carseats.test$Sales - pred.pruned)^2)
[1] 4.993124
```

Answer: No, increased instead.

(d)

```
> library(randomForest)
randomForest 4.6-12
Type rfNews() to see new features/changes/bug fixes.
> bag.carseats = randomForest(Sales ~ ., data = Carseats.train, mtry
= 10, ntree = 500,
+     importance = T)
> bag.pred = predict(bag.carseats, Carseats.test)
> mean((Carseats.test$Sales - bag.pred)^2)
[1] 2.604369
> importance(bag.carseats)
              %IncMSE IncNodePurity
CompPrice   14.4124562    133.731797
Income       6.5147532     74.346961
Advertising 15.7607104    117.822651
Population   0.6031237     60.227867
Price       57.8206926    514.802084
ShelveLoc   43.0486065    319.117972
Age         19.8789659    192.880596
Education    2.9319161     39.490093
Urban       -3.1300102      8.695529
US           7.6298722     15.723975
```

Answer:

Bagging improved the test MSE to 2.6. Price, ShelveLoc and Age are most important predictors of Sale.

(e)

```
> rf.carseats = randomForest(Sales ~ ., data = Carseats.train, mtry =
5, ntree = 500,
+     importance = T)
> rf.pred = predict(rf.carseats, Carseats.test)
mean((Carseats.test$Sales - rf.pred)^2)
> mean((Carseats.test$Sales - rf.pred)^2)
[1] 2.802383
> importance(rf.carseats)
              %IncMSE IncNodePurity
CompPrice   12.0259791     124.81403
Income       5.5542673     106.15418
Advertising 12.0466048     136.15204
Population   0.3136897      81.68162
Price       45.9639857     457.15711
```

```
ShelveLoc   36.2789679     271.76488
Age         20.8537727     196.72182
Education    2.9005332      54.16980
Urban       -0.6888196      11.86848
US           6.9739759      23.64075.
```

Answer:

Random forest worsens the MSE on the test set to 2.80. Price, ShelveLoc adn Age are three most important predictors of Sale.