Homework 4

Hyeonjin Lee

4/27/2021

Introduction to Statistical Learning in R Homework 4 (Chapters 5,6,7,8,9):

Chapter 5 Exercise 2

We will now derive the probability that a given observation is part of a bootstrap sample. Suppose that we obtain a bootstrap sample from a set of n observations.

a) What is the probability that the first bootstrap observation is not the jth observation from the original sample? Justify your answer.

The probability that the jth observation is selected as the first boostrap observation is 1/n. Therefore, the inverse would be 1 - 1/n.

b) What is the probability that the second bootstrap observation is not the jth observation from the original sample?

This is the same as part a (1 - 1/n) because the sampling is done with replacement.

c) Argue that the probability that the jth observation is not in the bootstrap sample is $(1-1/n)^n$

$$p_j(n) = \prod_{i=1}^{n} \pi_j = (1 - 1/n)^n$$

d) When n = 5, what is the probability that the jth observation is in the bootstrap sample? (1-(1-1/5)^5) = .672

e)) When n = 100, what is the probability that the jth observation is in the bootstrap sample?

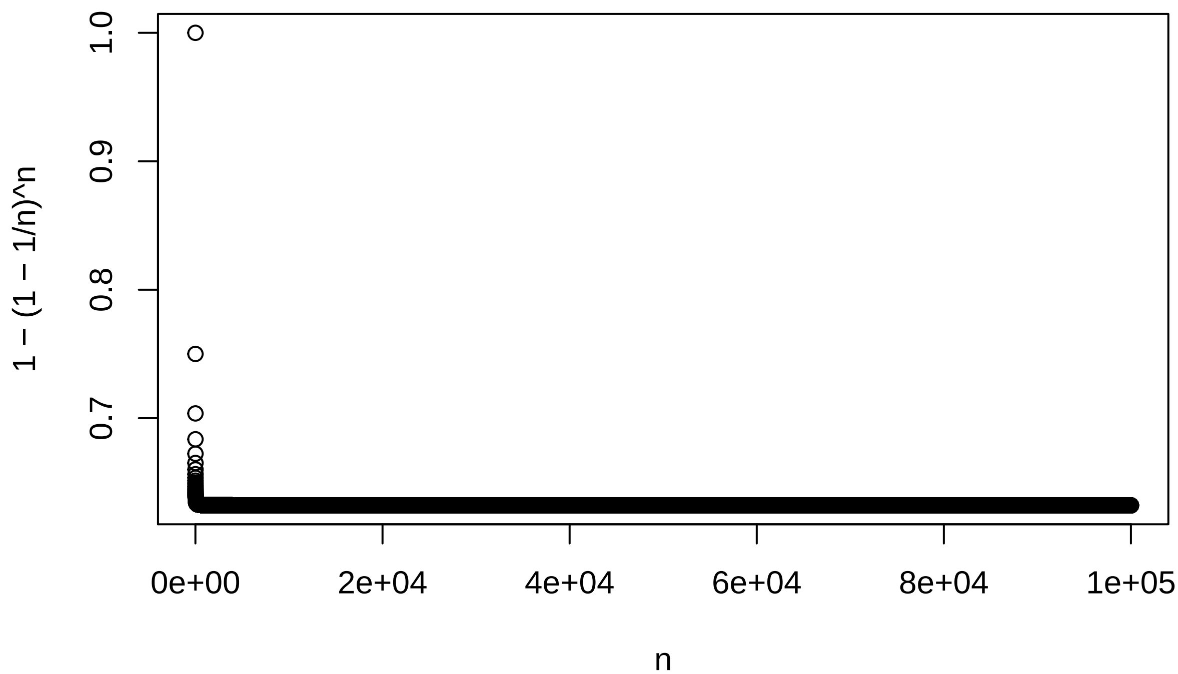
(1-(1-1/100)^100) = .634

f) When n = 10,000, what is the probability that the jth observation is in the bootstrap sample?

(1-(1/10000)^10000) = .632

g) Create a plot that displays, for each integer value of n from 1 to 100,000, the probability that the jth observation is in the bootstrap sample. Comment on what you observe.

```
n <- 1:100000
plot(n, 1 - (1 - 1/n)^n)
```



> Our graph lines up with our answers from previous parts as it reaches an asymptote of around .632.

h) We will now investigate numerically the probability that a bootstrap sample of size n = 100 contains the jth observation. Here j = 4. We repeatedly create bootstrap samples, and each time we record whether or not the fourth observation is contained in the bootstrap sample.

store=rep(NA, 10000)
for(i in 1:10000) {
store[i]=sum(sample (1:100, rep=TRUE)==4) >0
}
mean(store)

Comment on the results obtained.

```
store <- rep(NA, 10000)
for (i in 1:10000) {
    store[i] <- sum(sample(1:100, rep = TRUE) == 4) > 0
}
mean(store)
```

```
## [1] 0.6365
```

We are repeatedly creating bootstrap samples. We can see that all the observations converge to a value of around .632.

Chapter 5 Exercise 6

We continue to consider the use of a logistic regression model to predict the probability of default using income and balance on the Default data set. In particular, we will now compute estimates for the standard errors of the income and balance logistic regression coefficients in two different ways: (1) using the bootstrap, and (2) using the standard formula for computing the standard errors in the glm() function. Do not forget to set a random seed before beginning your analysis.

a) Using the summary() and glm() functions, determine the estimated standard errors for the coefficients associated with income and balance in a multiple logistic regression model that uses both predictors.

```
set.seed(1)
library(ISLR)
attach(Default)
fit.glm <- glm(default ~ income + balance, data = Default, family = "binomial")
summary(fit.glm)
```

```
##
## Call:
## glm(formula = default ~ income + balance, family = "binomial",
##     data = Default)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4725  -0.1444  -0.0574  -0.0211   3.7245
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.154e+01  4.348e-01 -26.545  < 2e-16 ***
## income       2.081e-05  4.985e-06   4.174 2.99e-05 ***
## balance      5.647e-03  2.274e-04  24.836  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2920.6  on 9999  degrees of freedom
## Residual deviance: 1579.0  on 9997  degrees of freedom
## AIC: 1585
##
## Number of Fisher Scoring iterations: 8
```

The standard errors of β_0,β_1,β_2 are 4.348e-01, 4.98e-06, and 2.274e-4.

b) Write a function, boot.fn(), that takes as input the Default data set as well as an index of the observations, and that outputs the coefficient estimates for income and balance in the multiple logistic regression model.

```
boot.fn <- function(data, index) {
    fit <- glm(default ~ income + balance, data = data, family = "binomial", subset = index)
    return (coef(fit))
}
```

c) Use the boot() function together with your boot.fn() function to estimate the standard errors of the logistic regression coefficients for income and balance.

```
# library(boot)
# set.seed(101, sample.kind = "Rounding")
#
# (results <- boot(data = Default, statistic = boot.fn, R = 1000))
```

The standard errors of β_0,β_1,β_2 are 4.24e-01, 4.58e-06, and 2.26e-4.

d) Comment on the estimated standard errors obtained using the glm() function and using your bootstrap function.

The standard deviations from part b and c are very similar.

Chapter 6 Exercise 9

In this exercise, we will predict the number of applications received using the other variables in the College data set.

(a) Split the data set into a training set and a test set.

```
library(ISLR)
data(College)
set.seed(11)
train = sample(1:dim(College)[1], dim(College)[1] / 2)
test <- -train
College.train <- College[train, ]
College.test <- College[test, ]
```

(b) Fit a linear model using least squares on the training set, and report the test error obtained.

```
fit.lm <- lm(Apps ~ ., data = College.train)
pred.lm <- predict(fit.lm, College.test)
mean((pred.lm - College.test$Apps)^2)
```

```
## [1] 1026096
```

(c) Fit a ridge regression model on the training set, with λ chosen by cross-validation. Report the test error obtained.

```
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 3.6.2
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-1
```

```r
train.mat <- model.matrix(Apps ~ ., data = College.train)
test.mat <- model.matrix(Apps ~ ., data = College.test)
grid <- 10 ^ seq(4, -2, length = 100)
fit.ridge <- glmnet(train.mat, College.train$Apps, alpha = 0, lambda = grid, thresh = 1e-12)
cv.ridge <- cv.glmnet(train.mat, College.train$Apps, alpha = 0, lambda = grid, thresh = 1e-12)
bestlam.ridge <- cv.ridge$lambda.min
bestlam.ridge
```

```
## [1] 0.01
```

```r
pred.ridge <- predict(fit.ridge, s = bestlam.ridge, newx = test.mat)
mean((pred.ridge - College.test$Apps)^2)
```

```
## [1] 1026069
```

(d) Fit a lasso model on the training set, with λ chosen by crossvalidation. Report the test error obtained, along with the number of non-zero coefficient estimates.

```r
fit.lasso <- glmnet(train.mat, College.train$Apps, alpha = 1, lambda = grid, thresh = 1e-12)
cv.lasso <- cv.glmnet(train.mat, College.train$Apps, alpha = 1, lambda = grid, thresh = 1e-12)
bestlam.lasso <- cv.lasso$lambda.min
bestlam.lasso
```

```
## [1] 0.01
```

```r
pred.lasso <- predict(fit.lasso, s = bestlam.lasso, newx = test.mat)
mean((pred.lasso - College.test$Apps)^2)
```

```
## [1] 1026036
```

```r
predict(fit.lasso, s = bestlam.lasso, type = "coefficients")
```

```
## 19 x 1 sparse Matrix of class "dgCMatrix"
##                         1
## (Intercept)   37.86520037
## (Intercept)    .
## PrivateYes  -551.14946609
## Accept         1.74980812
## Enroll        -1.36005786
## Top10perc     65.55655577
## Top25perc    -22.52640339
## F.Undergrad    0.10181853
## P.Undergrad    0.01789131
## Outstate      -0.08706371
## Room.Board     0.15384585
## Books         -0.12227313
## Personal       0.16194591
## PhD          -14.29638634
## Terminal      -1.03118224
## S.F.Ratio      4.47956819
## perc.alumni   -0.45456280
## Expend         0.05618050
## Grad.Rate      9.07242834
```

(e) Fit a PCR model on the training set, with M chosen by crossvalidation. Report the test error obtained, along with the value of M selected by cross-validation.

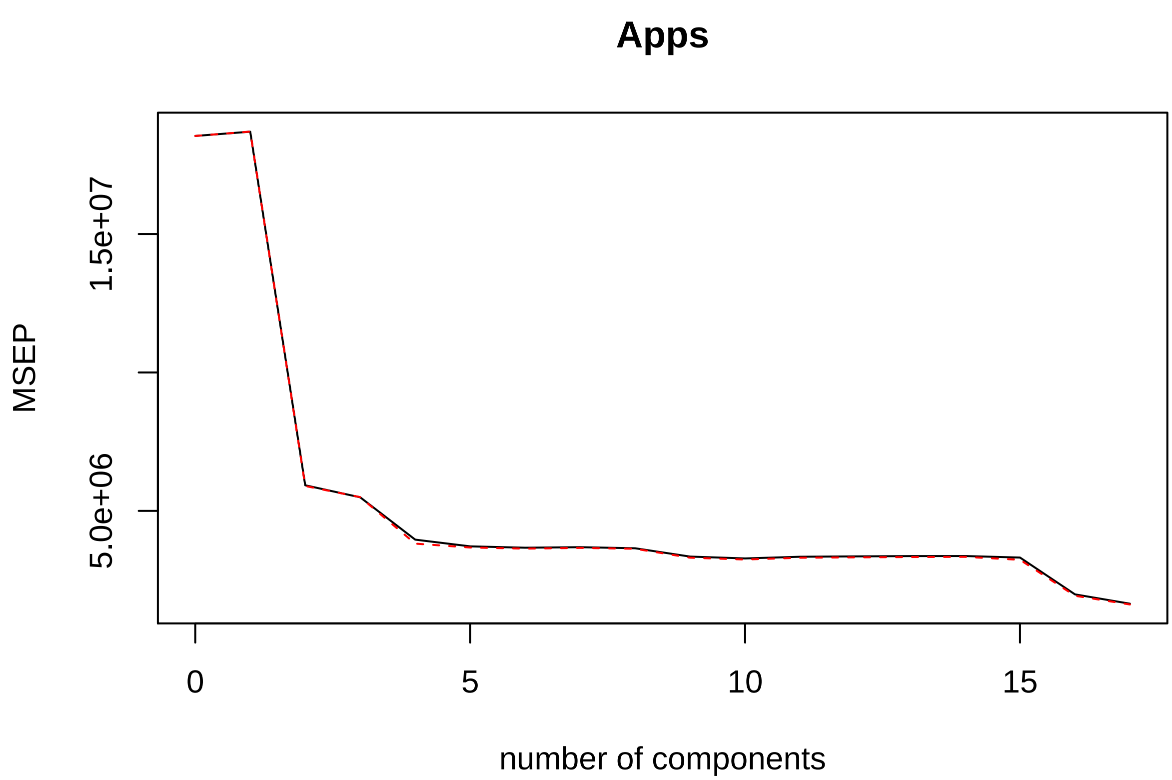
```
library(pls)
```

```
## Warning: package 'pls' was built under R version 3.6.2
```

```
##
## Attaching package: 'pls'
```

```
## The following object is masked from 'package:stats':
##
##     loadings
```

```
fit.pcr <- pcr(Apps ~ ., data = College.train, scale = TRUE, validation = "CV")
validationplot(fit.pcr, val.type = "MSEP")
```

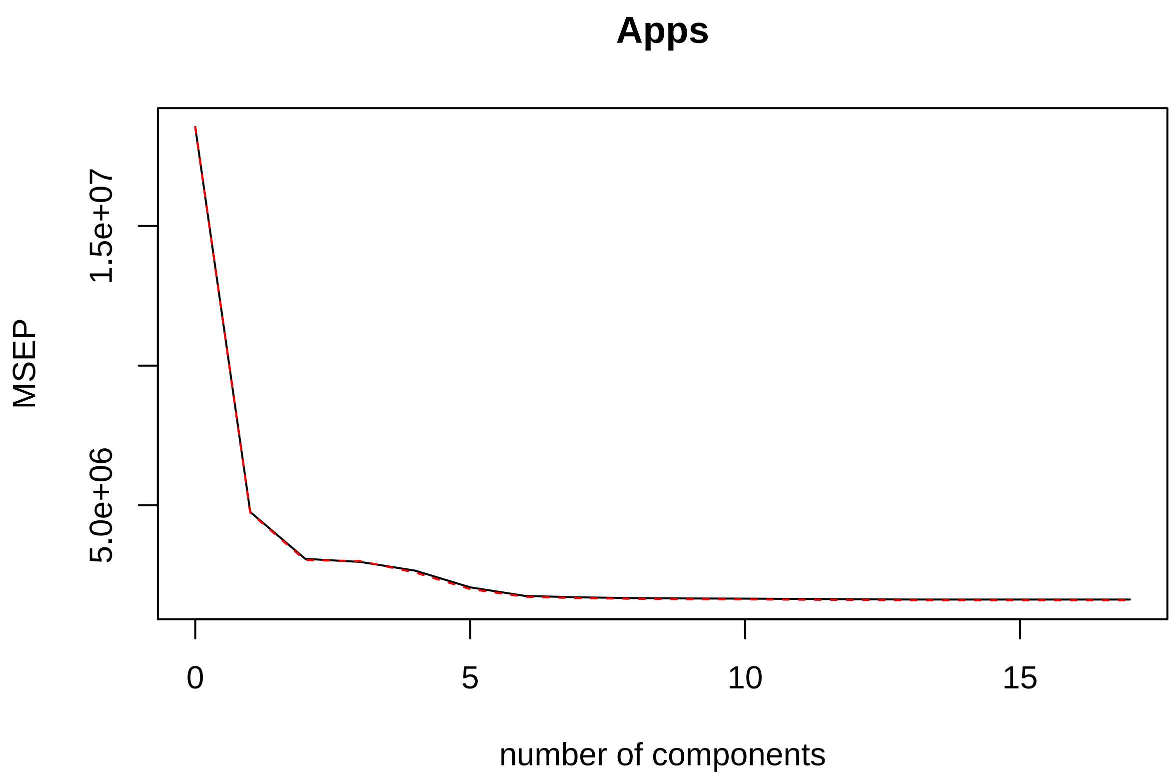



```
pred.pcr <- predict(fit.pcr, College.test, ncomp = 10)
mean((pred.pcr - College.test$Apps)^2)
```

```
## [1] 1867486
```

(f) Fit a PLS model on the training set, with M chosen by crossvalidation. Report the test error obtained, along with the value of M selected by cross-validation.

```r
fit.pls <- plsr(Apps ~ ., data = College.train, scale = TRUE, validation = "CV")
validationplot(fit.pls, val.type = "MSEP")
```



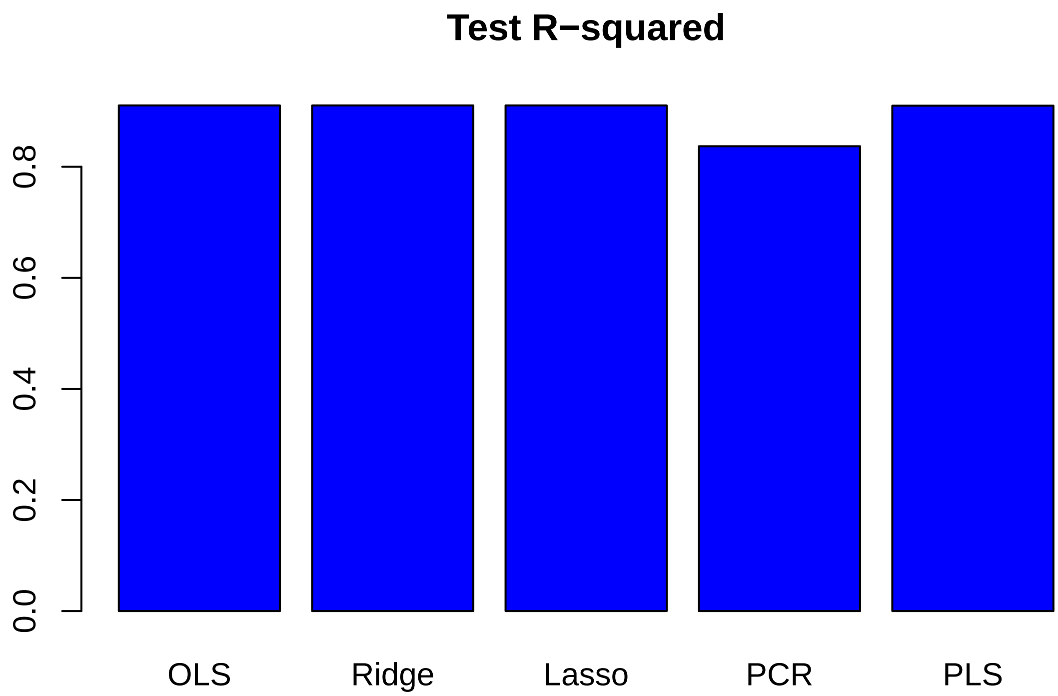

```r
pred.pls <- predict(fit.pls, College.test, ncomp = 10)
mean((pred.pls - College.test$Apps)^2)
```

```
## [1] 1031287
```

(g) Comment on the results obtained. How accurately can we predict the number of college applications received? Is there much difference among the test errors resulting from these five approaches?

```r
test.avg <- mean(College.test$Apps)
lm.r2 <- 1 - mean((pred.lm - College.test$Apps)^2) / mean((test.avg - College.test$Apps)^2)
ridge.r2 <- 1 - mean((pred.ridge - College.test$Apps)^2) / mean((test.avg - College.test$Apps)^2)
lasso.r2 <- 1 - mean((pred.lasso - College.test$Apps)^2) / mean((test.avg - College.test$Apps)^2)
pcr.r2 <- 1 - mean((pred.pcr - College.test$Apps)^2) / mean((test.avg - College.test$Apps)^2)
pls.r2 <- 1 - mean((pred.pls - College.test$Apps)^2) / mean((test.avg - College.test$Apps)^2)

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



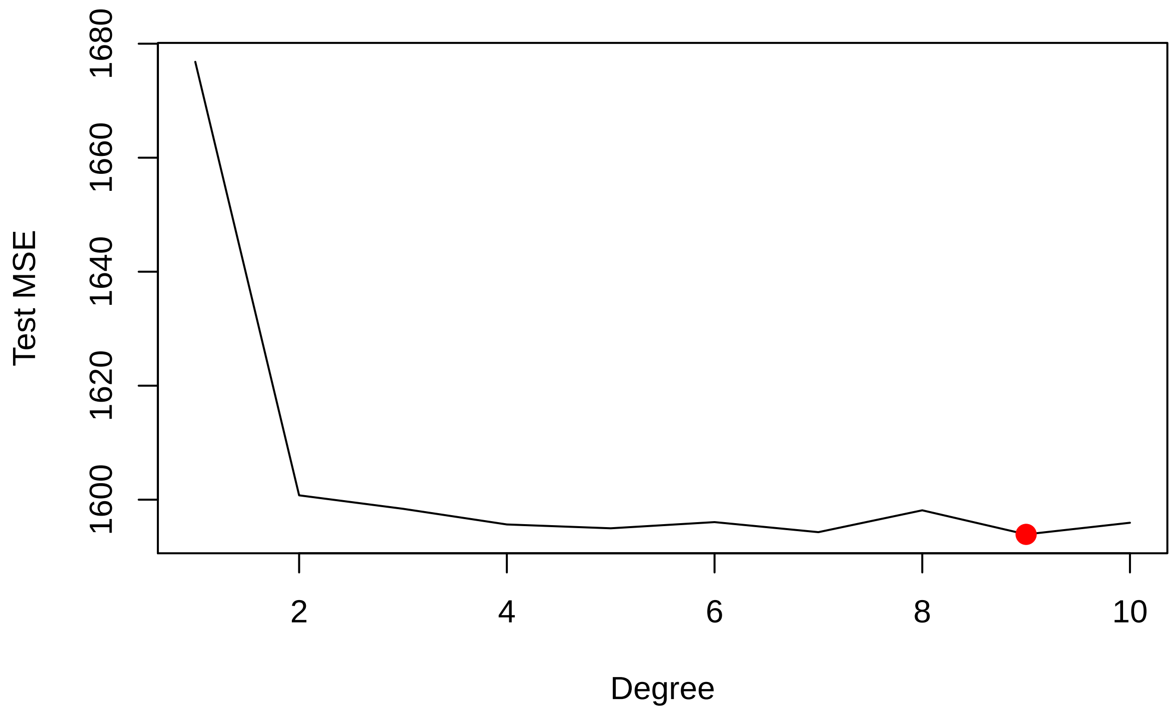
> The R^2 tests for all the plots are highly accurate and around .9 except for the PCR model.

Chapter 7 Exercise 6

In this exercise, you will further analyze the Wage data set considered throughout this chapter.

(a) Perform polynomial regression to predict wage using age. Use cross-validation to select the optimal degree d for the polynomial. What degree was chosen, and how does this compare to the results of hypothesis testing using ANOVA? Make a plot of the resulting polynomial fit to the data.

```
library(ISLR)
library(boot)
set.seed(1)
deltas <- rep(NA, 10)
for (i in 1:10) {
    fit <- glm(wage ~ poly(age, i), data = Wage)
    deltas[i] <- cv.glm(Wage, fit, K = 10)$delta[1]
}
plot(1:10, deltas, xlab = "Degree", ylab = "Test MSE", type = "l")
d.min <- which.min(deltas)
points(which.min(deltas), deltas[which.min(deltas)], col = "red", cex = 2, pch = 20)
```



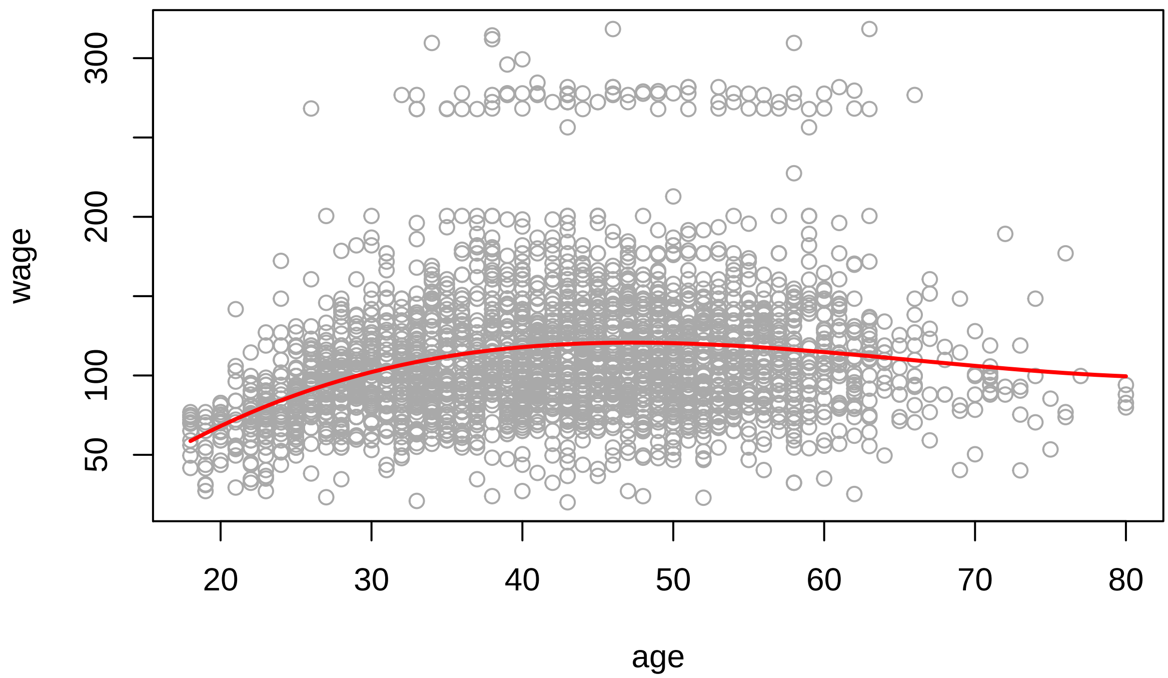
> The degree that was chosen was 4.

```
fit1 <- lm(wage ~ age, data = Wage)
fit2 <- lm(wage ~ poly(age, 2), data = Wage)
fit3 <- lm(wage ~ poly(age, 3), data = Wage)
fit4 <- lm(wage ~ poly(age, 4), data = Wage)
fit5 <- lm(wage ~ poly(age, 5), data = Wage)
anova(fit1, fit2, fit3, fit4, fit5)
```

```
## Analysis of Variance Table
##
## Model 1: wage ~ age
## Model 2: wage ~ poly(age, 2)
## Model 3: wage ~ poly(age, 3)
## Model 4: wage ~ poly(age, 4)
## Model 5: wage ~ poly(age, 5)
##   Res.Df     RSS Df Sum of Sq        F    Pr(>F)
## 1   2998 5022216
## 2   2997 4793430  1    228786 143.5931 < 2.2e-16 ***
## 3   2996 4777674  1     15756   9.8888  0.001679 **
## 4   2995 4771604  1      6070   3.8098  0.051046 .
## 5   2994 4770322  1      1283   0.8050  0.369682
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

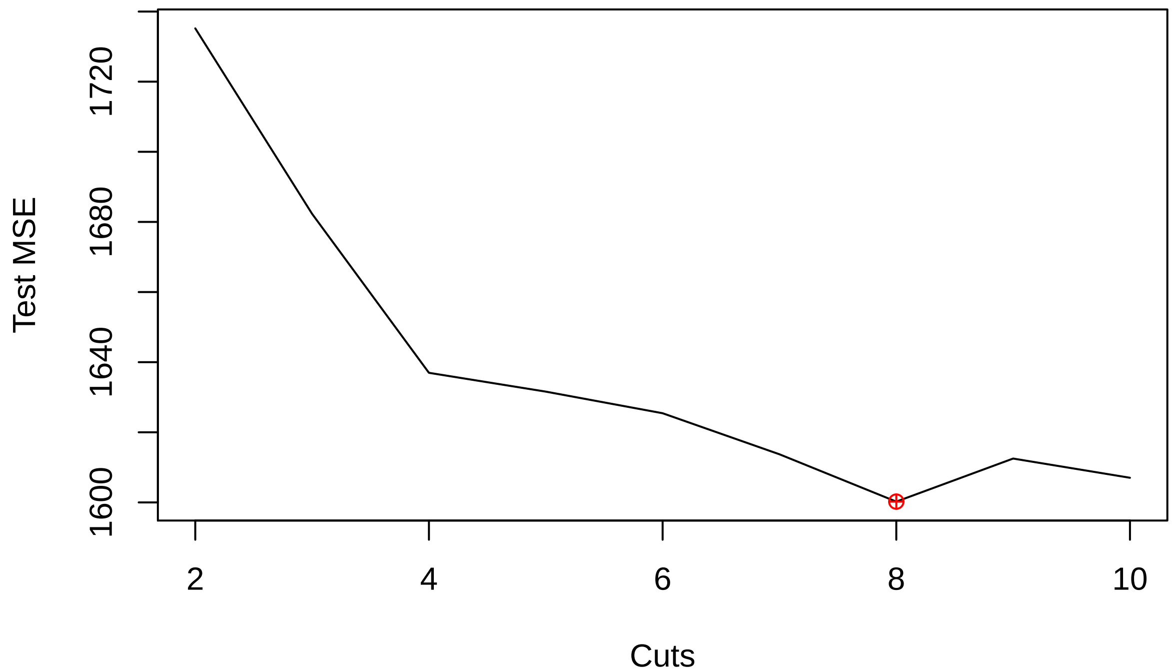
Looking at the p-values, we can see that cubic or quartic polynomial appear to be a reasonable fit to the data.

```r
plot(wage ~ age, data = Wage, col = "darkgrey")
agelims <- range(Wage$age)
age.grid <- seq(from = agelims[1], to = agelims[2])
fit <- lm(wage ~ poly(age, 3), data = Wage)
preds <- predict(fit, newdata = list(age = age.grid))
lines(age.grid, preds, col = "red", lwd = 2)
```

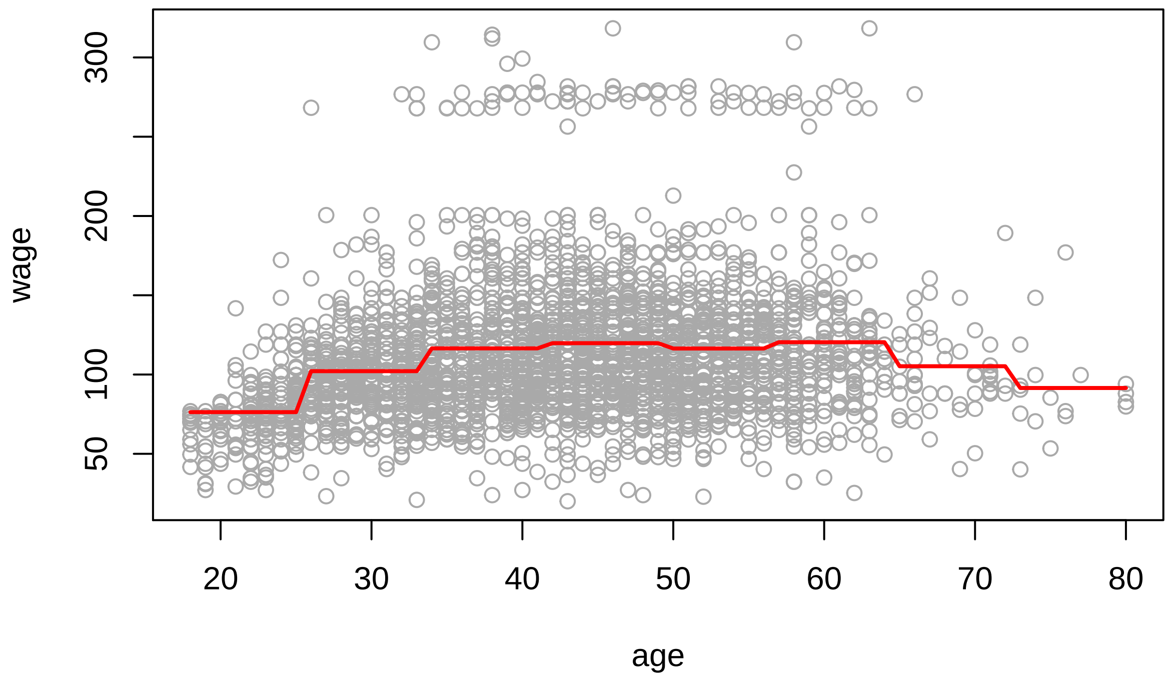


(b) Fit a step function to predict wage using age, and perform crossvalidation to choose the optimal number of cuts. Make a plot of the fit obtained.

```r
cvs <- rep(NA, 10)
for (i in 2:10) {
    Wage$age.cut <- cut(Wage$age, i)
    fit <- glm(wage ~ age.cut, data = Wage)
    cvs[i] <- cv.glm(Wage, fit, K = 10)$delta[1]
}
plot(2:10, cvs[-1], xlab = "Cuts", ylab = "Test MSE", type = "l")
d.min <- which.min(cvs)
points(which.min(cvs), cvs[which.min(cvs)], col = "red", cex = 1, pch = 10)
```




```r
plot(wage ~ age, data = Wage, col = "darkgrey")
agelims <- range(Wage$age)
age.grid <- seq(from = agelims[1], to = agelims[2])
fit <- glm(wage ~ cut(age, 8), data = Wage)
preds <- predict(fit, data.frame(age = age.grid))
lines(age.grid, preds, col = "red", lwd = 2)
```



Chapter 8 Exercise 8

In the lab, a classification tree was applied to the Carseats data set after converting Sales into a qualitative response variable. Now we will seek to predict Sales using regression trees and related approaches, treating the response as a quantitative variable.

(a) Split the data set into a training set and a test set.

```
library(ISLR)
attach(Carseats)
set.seed(1)
subset<-sample(nrow(Carseats),nrow(Carseats)*0.7)
car.train<-Carseats[subset,]
car.test<-Carseats[-subset,]
```

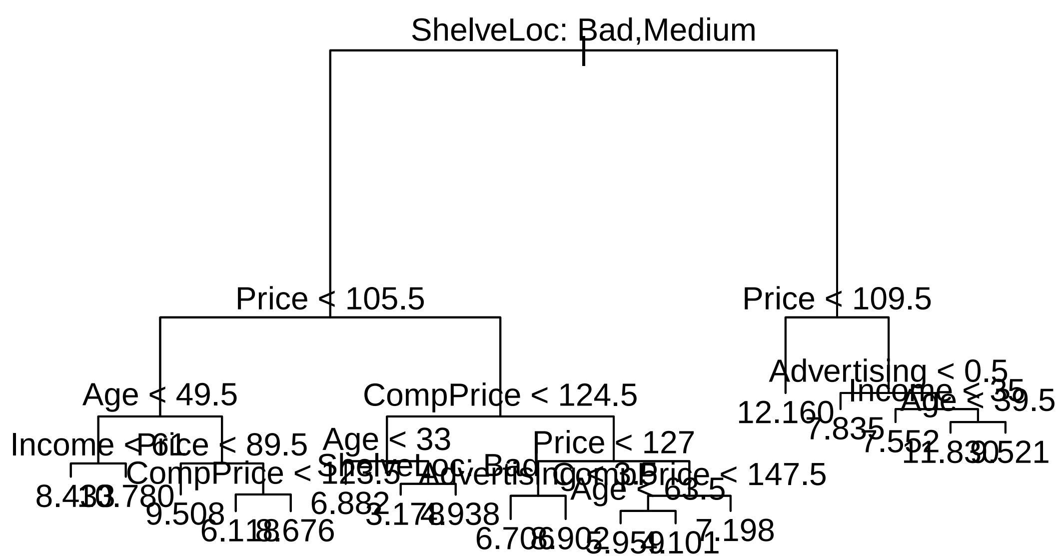
(b) Fit a regression tree to the training set. Plot the tree, and interpret the results. What test MSE do you obtain?

```
library(tree)
car.model.train<-tree(Sales~.,car.train)
summary(car.model.train)
```

```
##
```

```
## Regression tree:
## tree(formula = Sales ~ ., data = car.train)
## Variables actually used in tree construction:
## [1] "ShelveLoc"   "Price"       "Age"         "Income"      "CompPrice"
## [6] "Advertising"
## Number of terminal nodes:  18
## Residual mean deviance:  2.409 = 631.1 / 262
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -4.77800 -0.96100 -0.08865  0.00000  1.01800  4.14100
```

```r
plot(car.model.train)
text(car.model.train,pretty=0)
```



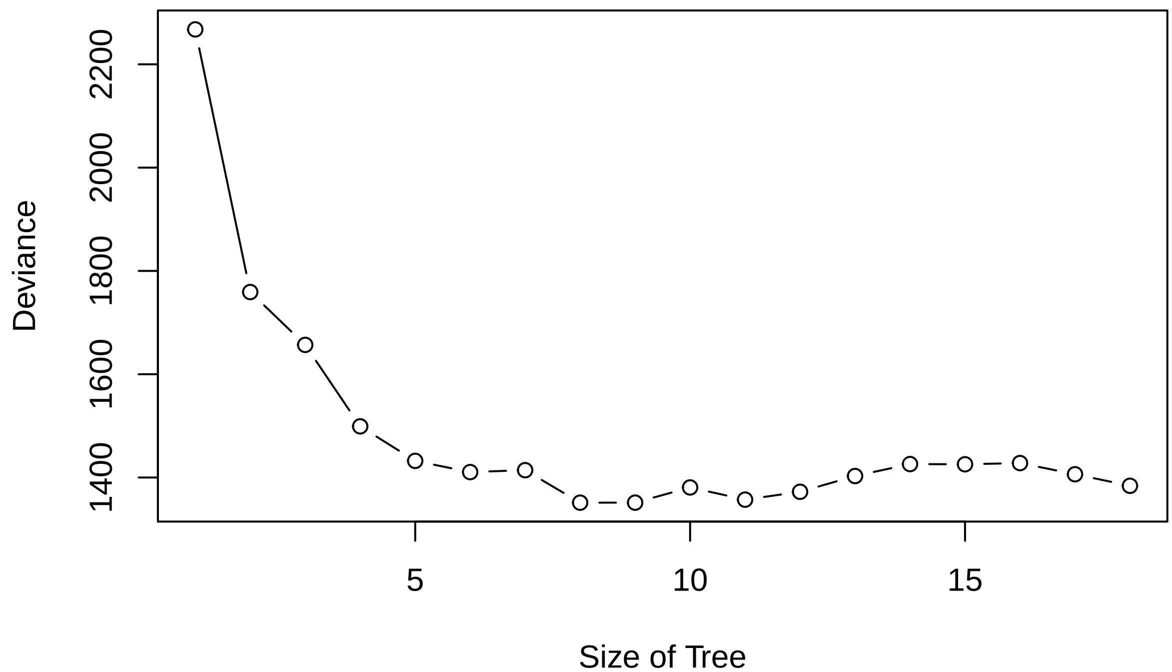

```r
tree.prediction<-predict(car.model.train,newdata=car.test)
tree.mse<-mean((car.test$Sales-tree.prediction)^2)
tree.mse
```

```
## [1] 4.208383
```

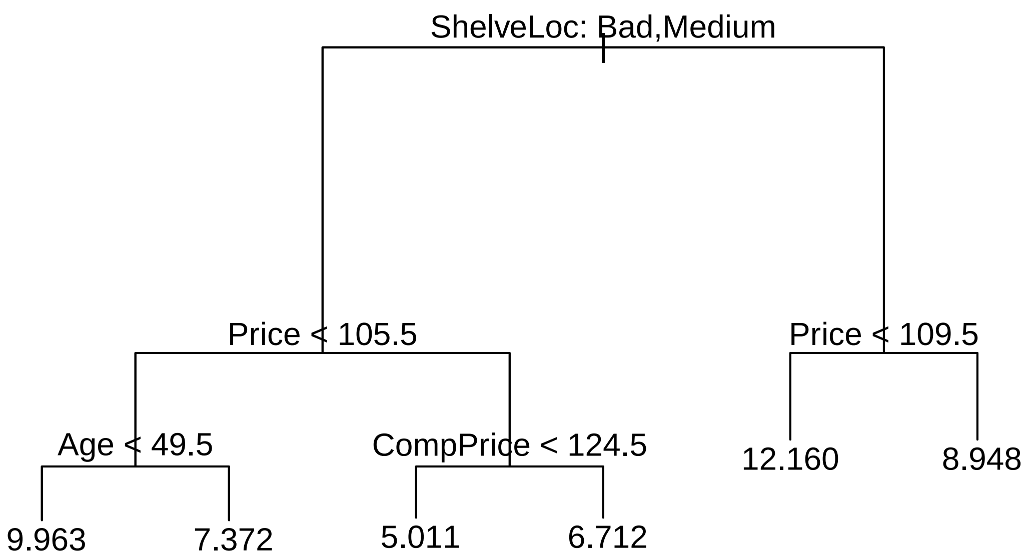
We can see that the test MSE for the tree is 2.588

(c) Use cross-validation in order to determine the optimal level of tree complexity. Does pruning the tree improve the test MSE?

```
set.seed(1)
cv.car<-cv.tree(car.model.train)
plot(cv.car$size,cv.car$dev,xlab = "Size of Tree",ylab = "Deviance",type = "b")
```




```
prune.car<-prune.tree(car.model.train,best=6)
plot(prune.car)
text(prune.car,pretty=0)
```




```
prune.predict<-predict(prune.car,car.test)
mean((prune.predict-car.test$Sales)^2)
```

```
## [1] 5.118217
```

We can see that pruning increased the MSE to 5.4538

(d) Use the bagging approach in order to analyze this data. What test MSE do you obtain? Use the importance() function to determine which variables are most important.

```
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
bag.car<-randomForest(Sales~.,car.train,importance=TRUE,mtry=13)
```

```
## Warning in randomForest.default(m, y, ...): invalid mtry: reset to within
## valid range
```

```r
importance(bag.car)
```

```
##                %IncMSE IncNodePurity
## CompPrice   34.6322504     233.60705
## Income       5.3645204     116.93827
## Advertising 18.8175105     153.05938
## Population  -3.0858810      64.26621
## Price       70.9386948     698.15948
## ShelveLoc   74.2328945     645.49148
## Age         20.8561302     224.71954
## Education    1.3723565      61.47839
## Urban       -1.9986734      10.51832
## US           0.8095402      10.19895
```

```r
bag.car.predict<-predict(bag.car,car.test)
mean((bag.car.predict-car.test$Sales)^2)
```

```
## [1] 2.571169
```

The test MSE is 2.351882. We can see that the two most important variables are Price and ShelveLoc.

(e) Use random forests to analyze this data. What test MSE do you obtain? Use the importance() function to determine which variables are most important. Describe the effect of m, the number of variables considered at each split, on the error rate obtained.

```r
rf.car<-randomForest(Sales~.,car.train,importance=TRUE,mtry=sqrt(13))
importance(rf.car)
```

```
##                %IncMSE IncNodePurity
## CompPrice   20.7300392     213.78497
## Income       3.5122804     148.32279
## Advertising 15.6121947     159.16307
## Population   0.5759461     113.69354
## Price       51.9015680     594.84872
## ShelveLoc   51.4866473     539.23503
## Age         18.6833946     261.97525
## Education    3.0894573      88.05427
## Urban       -2.4726183      17.29229
## US           4.0933782      27.11808
```

```r
rf.car.predict<-predict(rf.car,car.test)
mean((rf.car.predict-car.test$Sales)^2)
```

```
## [1] 2.674922
```

The most important variables are also Price and ShelveLoc. Random forest avoids correlated trees and should perform better than Bagging however it is not the case here. m = p = 13.

Chapter 9 Exercise 8

This problem involves the OJ data set which is part of the ISLR package.

(a) Create a training set containing a random sample of 800 observations, and a test set containing the remaining observations.

```
set.seed(1)
train <- sample(nrow(OJ), 800)
test <- -train
train_OJ <- OJ[train,]
test_OJ <- OJ[test,]
```

(b) Fit a support vector classifier to the training data using cost=0.01, with Purchase as the response and the other variables as predictors. Use the summary() function to produce summary statistics, and describe the results obtained.

```
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 3.6.2
```

```
OJ$STORE = as.factor(OJ$STORE)
OJ$Store7 = NULL
OJ$StoreID = NULL
OJ.train = OJ[train,]
OJ.test = OJ[-train,]
svm1 = svm(Purchase~.,data=OJ.train,kernel='linear',cost=0.01)
summary(svm1)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = OJ.train, kernel = "linear",
##     cost = 0.01)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  0.01
##
## Number of Support Vectors:  439
##
##  ( 219 220 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

The model selects 434 out of 800 observations as supportp oints. We are predicting 2 classes.

(c) What are the training and test error rates?

```
svm.train.pred <- predict(svm1, OJ.train)
svm.test.pred <- predict(svm1, OJ.test)
train.error <- mean(svm.train.pred != OJ.train$Purchase)
test.error <- mean(svm.test.pred != OJ.test$Purchase)
summary(train.error)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.17    0.17    0.17    0.17    0.17    0.17
```

```
summary(test.error)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1593  0.1593  0.1593  0.1593  0.1593  0.1593
```

The train error is .16 and the test error is .1889.

(d) Use the tune() function to select an optimal cost. Consider values in the range 0.01 to 10.

```
svm1.tune = tune(svm,Purchase~.,data=OJ.train, ranges=list(cost=c(.01,.02,.05,.1,.2,.5,1,2,5,10)),kernel
summary(svm1.tune)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##  0.02
##
## - best performance: 0.17375
##
## - Detailed performance results:
##     cost   error dispersion
## 1   0.01 0.18250 0.03129164
## 2   0.02 0.17375 0.03197764
## 3   0.05 0.17375 0.03508422
## 4   0.10 0.17625 0.02791978
## 5   0.20 0.17875 0.03335936
## 6   0.50 0.17875 0.03438447
## 7   1.00 0.17875 0.03438447
## 8   2.00 0.17875 0.03120831
## 9   5.00 0.17500 0.03435921
## 10 10.00 0.17625 0.03701070
```

(e) Compute the training and test error rates using this new value for cost.

```r
svm.bestFit <- svm(Purchase ~., data = OJ.train, kernel = "linear", cost = 1)
svm.train.pred2 <- predict(svm.bestFit, OJ.train)
svm.test.pred2 <- predict(svm.bestFit, OJ.test)
train.error2 <- mean(svm.train.pred2 != OJ.train$Purchase)
test.error2 <- mean(svm.test.pred2 != OJ.test$Purchase)

summary(train.error2)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1663  0.1663  0.1663  0.1663  0.1663  0.1663
```

```r
summary(test.error2)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1556  0.1556  0.1556  0.1556  0.1556  0.1556
```

The train error is .16 and the test error is .1815.

(f) Repeat parts (b) through (e) using a support vector machine with a radial kernel. Use the default value for gamma.

```r
svm.fit <- svm(Purchase ~., data = OJ.train, kernel = "radial", cost = 0.01)
summary(svm.fit)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = OJ.train, kernel = "radial",
##     cost = 0.01)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  0.01
##
## Number of Support Vectors:  632
##
##  ( 317 315 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

```r
svm.train.pred <- predict(svm.fit, OJ.train)
svm.test.pred <- predict(svm.fit, OJ.test)
train.error <- mean(svm.train.pred != OJ.train$Purchase)
test.error <- mean(svm.test.pred != OJ.test$Purchase)

summary(train.error)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.3937  0.3937  0.3937  0.3937  0.3937  0.3937
```

```r
summary(test.error)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.3778  0.3778  0.3778  0.3778  0.3778  0.3778
```

The train error is .3825 and the test error is .4111.

```r
svm.tune <- tune(svm, Purchase ~ ., data = OJ.train, kernel = "radial", ranges = list(cost = c(0.01, 0.
summary.tune <- summary(svm.tune)
summary.tune
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     1
##
## - best performance: 0.1725
##
## - Detailed performance results:
##    cost   error dispersion
## 1  0.01 0.39375 0.06568284
## 2  0.10 0.17500 0.04965156
## 3  1.00 0.17250 0.04518481
## 4  5.00 0.18500 0.05163978
## 5 10.00 0.18625 0.05118390
```

The best performance was at .1725.

```r
svm.bestFit <- svm(Purchase ~., data = OJ.train, kernel = "radial", cost = 5)
svm.train.pred <- predict(svm.bestFit, OJ.train)
svm.test.pred <- predict(svm.bestFit, OJ.test)
train.error <- mean(svm.train.pred != OJ.train$Purchase)
test.error <- mean(svm.test.pred != OJ.test$Purchase)

summary(train.error)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.145   0.145   0.145   0.145   0.145   0.145
```

```r
summary(test.error)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1852  0.1852  0.1852  0.1852  0.1852  0.1852
```

The new train error is .1375 and the new test error is .1778.

(g) Repeat parts (b) through (e) using a support vector machine with a polynomial kernel. Set degree=2.

```
svm.fit <- svm(Purchase ~., data = OJ.train, kernel = "polynomial", cost = 0.01, degree=2)
summary(svm.fit)
```

```
##
## Call:
## svm(formula = Purchase ~ ., data = OJ.train, kernel = "polynomial",
##     cost = 0.01, degree = 2)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  0.01
##      degree:  2
##      coef.0:  0
##
## Number of Support Vectors:  632
##
##  ( 317 315 )
##
##
## Number of Classes:  2
##
## Levels:
##  CH MM
```

```
svm.train.pred <- predict(svm.fit, OJ.train)
svm.test.pred <- predict(svm.fit, OJ.test)
train.error <- mean(svm.train.pred != OJ.train$Purchase)
test.error <- mean(svm.test.pred != OJ.test$Purchase)

summary(train.error)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.3937  0.3937  0.3937  0.3937  0.3937  0.3937
```

```
summary(test.error)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.3778  0.3778  0.3778  0.3778  0.3778  0.3778
```

The train error is .3825 and the test error is .4111.

```
svm.tune <- tune(svm, Purchase ~ ., data = OJ.train, kernel = "polynomial", ranges = list(cost = c(0.01
summary.tune <- summary(svm.tune)
summary.tune
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##    10
##
## - best performance: 0.18375
##
## - Detailed performance results:
##    cost   error dispersion
## 1  0.01 0.39375 0.08191501
## 2  0.10 0.34500 0.06015027
## 3  1.00 0.21875 0.07174656
## 4  5.00 0.19250 0.05210833
## 5 10.00 0.18375 0.04752558
```

The best performance was at .1775.

```r
svm.bestFit <- svm(Purchase ~., data = OJ.train, kernel = "polynomial", cost = 10)
svm.train.pred <- predict(svm.bestFit, OJ.train)
svm.test.pred <- predict(svm.bestFit, OJ.test)
train.error <- mean(svm.train.pred != OJ.train$Purchase)
test.error <- mean(svm.test.pred != OJ.test$Purchase)

summary(train.error)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.155   0.155   0.155   0.155   0.155   0.155
```

```r
summary(test.error)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.2111  0.2111  0.2111  0.2111  0.2111  0.2111
```

The new train error is .1537 and the new test error is .2037.

(h) Overall, which approach seems to give the best results on this data?

Comparing all the results, the linear kernel test performs the best.