School of Computing and Engineering: University of Missouri – Kansas City

Course Name: COMP_SCI-5565-0002 Intro to Statistical Learning

Instructor: Alexa Summers

Group Name: StatLearning Geeks

Group Members:

Bharath Kumar Pasunoori

Rakesh Rao Thakkalapelly

Pavan Kumar Kopera

Swathi Kunduru

Lakshmi Prasanna Peddi

FINAL EXAM

GitHub Link: https://github.com/thakkalapellyrakesh/ISL-Final-Exam.git

Chapter 8 Conceptual Problem 5:

Suppose we produce ten bootstrapped samples from a data set containing red and green classes. We then apply a classification tree to each bootstrapped sample and, for a specific value of X, produce 10 estimates of P(Class is Red|X):

0.1, 0.15, 0.2, 0.2, 0.55, 0.6, 0.6, 0.65, 0.7, and 0.75.

There are two common ways to combine these results together into a single class prediction. One is the majority vote approach discussed in this chapter. The second approach is to classify based on the average probability. In this example, what is the final classification under each of these two approaches?

```
> prob <- c(0.1, 0.15, 0.2, 0.2, 0.55, 0.6, 0.6, 0.65, 0.7, 0.75)
> #Majority Vote:
> sum(prob >= 0.5) # number of 'Red' predictions
[1] 6
> sum(prob < 0.5) # number of 'Green' predictions
[1] 4
> ifelse(sum(prob >= 0.5) > sum(prob < 0.5), "Red", "Green")
[1] "Red"
> #Average Approach:
> mean(prob) # average P(Red)
[1] 0.45
> #Average Approach:
> mean(prob) # average P(Red)
[1] 0.45
> ifelse(mean(prob) >= 0.5, "Red", "Green")
[1] "Green"
```

With the majority vote approach, we classify X as Red as it is the most commonly occurring class among the 10 predictions (6 for Red vs 4 for Green). With the average probability approach, we classify X as Green as the average of the 10 probabilities is 0.45.

Chapter 8 Applied Problem 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.

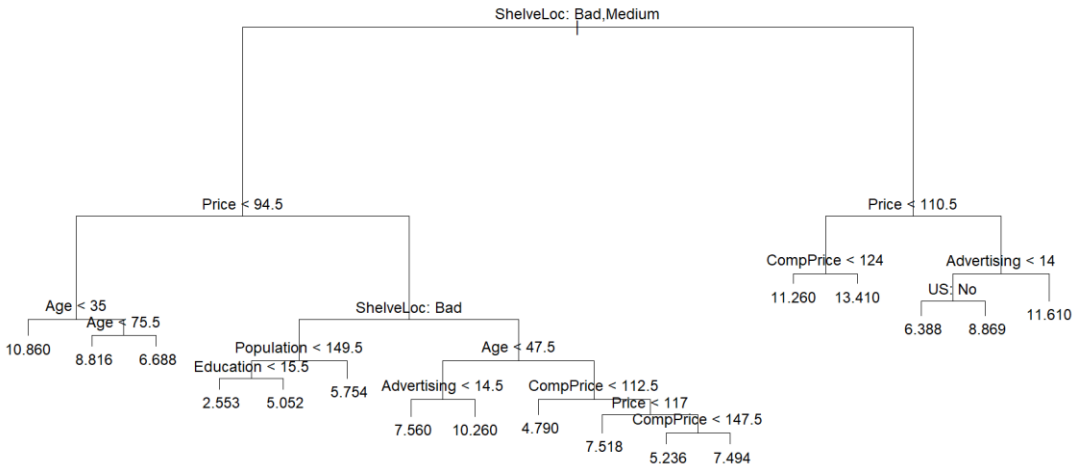
(a) Split the data set into a training set and a test set.

```
> #Chapter 8 Applied Problem 8:
> #a)
> install.packages("ISLR")
WARNING: Rtools is required to build R packages but is not currently installed. Please download and instal
l the appropriate version of Rtools before proceeding:

https://cran.rstudio.com/bin/windows/Rtools/
Installing package into 'C:/Users/pasun/AppData/Local/R/win-library/4.3'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.3/ISLR_1.4.zip'
Content type 'application/zip' length 2924200 bytes (2.8 MB)
downloaded 2.8 MB

package 'ISLR' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\pasun\AppData\Local\Temp\RtmpKU8yMw\downloaded_packages
> library(ISLR)

Attaching package: 'ISLR'

The following object is masked _by_ '.GlobalEnv':

    Auto

Warning message:
package 'ISLR' was built under R version 4.3.1
> attach(Carseats)
> train <- sample(1:nrow(Carseats), nrow(Carseats) / 2)
> Carseats.train <- Carseats[train, ]
> Carseats.test <- Carseats[-train, ]
>
```

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

```
> #b)
> install.packages("tree")
WARNING: Rtools is required to build R packages but is not currently installed. Please download and instal
l the appropriate version of Rtools before proceeding:

https://cran.rstudio.com/bin/windows/Rtools/
Installing package into 'C:/Users/pasun/AppData/Local/R/win-library/4.3'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.3/tree_1.0-43.zip'
Content type 'application/zip' length 159083 bytes (155 KB)
downloaded 155 KB

package 'tree' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\pasun\AppData\Local\Temp\RtmpKU8yMw\downloaded_packages
> library(tree)
Warning message:
package 'tree' was built under R version 4.3.1
```

```
> 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"         "Population"  "Education"   "Advertising" "CompPrice"
[8] "US"
Number of terminal nodes:  17
Residual mean deviance:  2.413 = 441.6 / 183
Distribution of residuals:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-4.6300 -1.0280  0.0015  0.0000  0.9552  4.3860
> plot(tree.carseats)
> text(tree.carseats, pretty = 0)
```

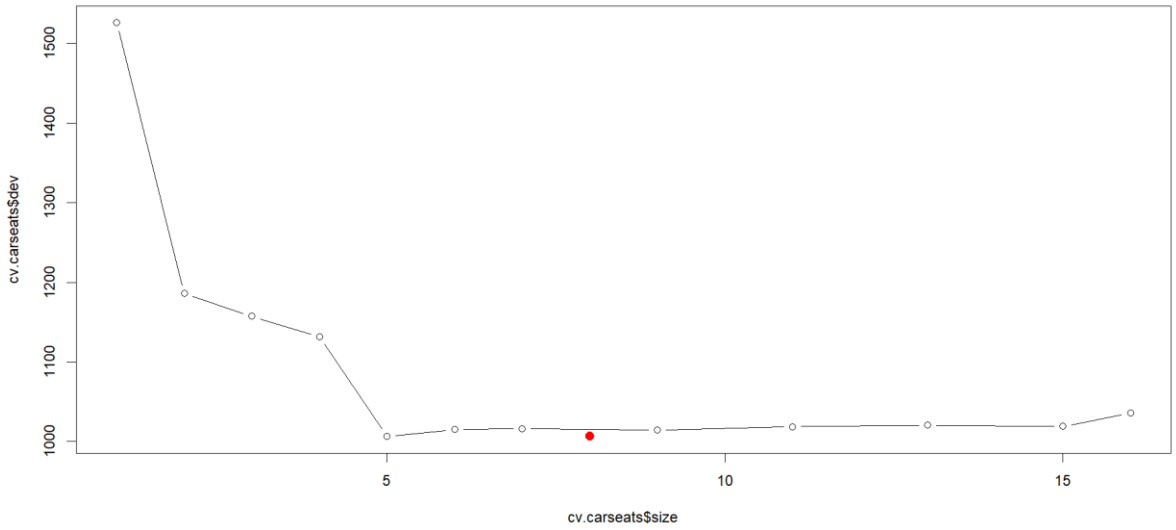



```
> yhat <- predict(tree.carseats, newdata = Carseats.test)
> mean((yhat - Carseats.test$Sales)^2)
[1] 4.371771
```

We may conclude that the Test MSE is about 4.37.

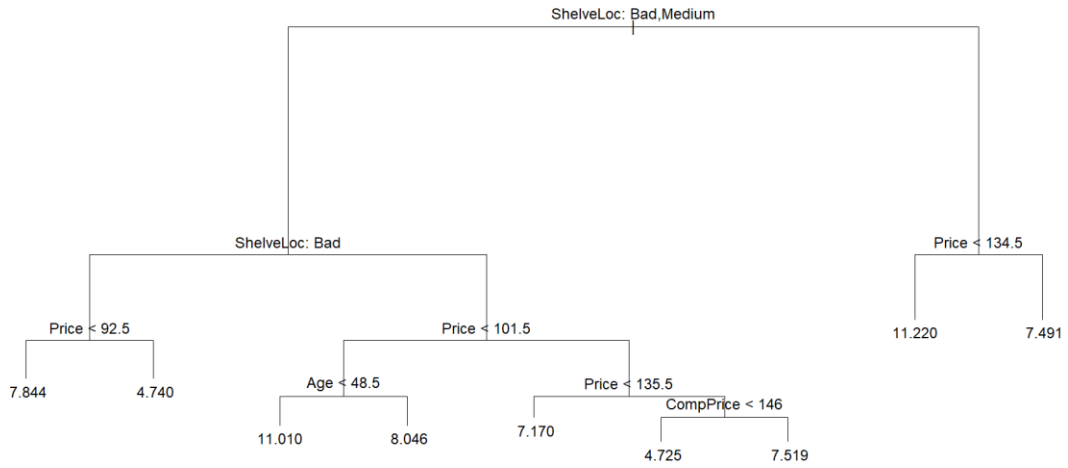
(c) Use cross-validation in order to determine the optimal level of tree complexity. Does pruning the tree improve the test MSE?

```
> #c)
> cv.carseats <- cv.tree(tree.carseats)
> plot(cv.carseats$size, cv.carseats$dev, type = "b")
> tree.min <- which.min(cv.carseats$dev)
> points(tree.min, cv.carseats$dev[tree.min], col = "red", cex = 2, pch = 20)
>
```



In this case, the tree of size 8 is selected by cross-validation. We now prune the tree to obtain the 8-node tree.

```
> prune.carseats <- prune.tree(tree.carseats, best = 8)
> plot(prune.carseats)
> text(prune.carseats, pretty = 0)
```




```
> yhat <- predict(prune.carseats, newdata = Carseats.test)
> mean((yhat - Carseats.test$Sales)^2)
[1] 5.075903
```

We may see that pruning the tree increases the Test MSE to 5.1.

(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.

```
> #d)
> install.packages("randomForest")
WARNING: Rtools is required to build R packages but is not currently installed. Please download and instal
l the appropriate version of Rtools before proceeding:

https://cran.rstudio.com/bin/windows/Rtools/
Installing package into 'C:/Users/pasun/AppData/Local/R/win-library/4.3'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.3/randomForest_4.7-1.1.zip'
Content type 'application/zip' length 222332 bytes (217 KB)
downloaded 217 KB

package 'randomForest' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\pasun\AppData\Local\Temp\RtmpGAXddF\downloaded_packages
> library(randomForest)
randomForest 4.7-1.1
Type rfNews() to see new features/changes/bug fixes.
Warning message:
package 'randomForest' was built under R version 4.3.1
>
```

```
> bag.carseats <- randomForest(Sales ~ ., data = Carseats.train, mtry = 10, ntree = 500, importance = TRU
E)
> yhat.bag <- predict(bag.carseats, newdata = Carseats.test)
> mean((yhat.bag - Carseats.test$Sales)^2)
[1] 2.758793
>
```

We may see that bagging decreases the Test MSE to 2.7.

```
> importance(bag.carseats)
              %IncMSE IncNodePurity
CompPrice   20.724998    130.421567
Income       2.616103     66.153373
Advertising 14.214948    121.847519
Population  -1.433690     58.523885
Price       50.544432    408.079671
ShelveLoc   57.131042    495.464946
Age         14.442394    118.497755
Education   -1.849036     38.905898
Urban       -3.593040      7.957335
US           5.850580     11.115617
>
```

We may conclude that "Price" and "ShelveLoc" are the two most important variables.

(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.

```
> #e)
> rf.carseats <- randomForest(Sales ~ ., data = Carseats.train, mtry = 3, ntree = 500, importance = TRUE)
> yhat.rf <- predict(rf.carseats, newdata = Carseats.test)
> mean((yhat.rf - Carseats.test$Sales)^2)
[1] 3.36742
```

In this case, with m=√p, we have a Test MSE of 3.4.

```
> importance(rf.carseats)
                %IncMSE IncNodePurity
CompPrice    9.13206555     127.87589
Income       0.55571004     106.44280
Advertising 13.42575151     170.45304
Population  -0.09152891      97.55192
Price       31.14563456     324.35874
ShelveLoc   37.35563450     348.94231
Age          8.85914380     131.46274
Education   -0.48270002      61.54373
Urban        0.48953546      14.00106
US           7.55566167      33.73913
>
```

We may conclude that, in this case also, “Price” and “ShelveLoc” are the two most important variables.

Chapter 8 Applied Problem 10:

We now use boosting to predict Salary in the Hitters data set.

(a) Remove the observations for whom the salary information is unknown, and then log-transform the salaries.

```
> #Chapter 8 Applied Problem 10:
> #a)
> attach(Hitters)
> Hitters<-na.omit(Hitters)
> Hitters$Salary<-log(Hitters$Salary)
```

```
> summary(Hitters)
     AtBat            Hits           HmRun            Runs             RBI             Walks       
 Min.   : 19.0   Min.   :  1.0   Min.   : 0.00   Min.   :  0.00   Min.   :  0.00   Min.   :  0.00  
 1st Qu.:282.5   1st Qu.: 71.5   1st Qu.: 5.00   1st Qu.: 33.50   1st Qu.: 30.00   1st Qu.: 23.00  
 Median :413.0   Median :103.0   Median : 9.00   Median : 52.00   Median : 47.00   Median : 37.00  
 Mean   :403.6   Mean   :107.8   Mean   :11.62   Mean   : 54.75   Mean   : 51.49   Mean   : 41.11  
 3rd Qu.:526.0   3rd Qu.:141.5   3rd Qu.:18.00   3rd Qu.: 73.00   3rd Qu.: 71.00   3rd Qu.: 57.00  
 Max.   :687.0   Max.   :238.0   Max.   :40.00   Max.   :130.00   Max.   :121.00   Max.   :105.00  
     Years            CAtBat           CHits            CHmRun           CRuns       
 Min.   : 1.000   Min.   :   19.0   Min.   :   4.0   Min.   :  0.00   Min.   :   2.0  
 1st Qu.: 4.000   1st Qu.:  842.5   1st Qu.: 212.0   1st Qu.: 15.00   1st Qu.: 105.5  
 Median : 6.000   Median : 1931.0   Median : 516.0   Median : 40.00   Median : 250.0  
 Mean   : 7.312   Mean   : 2657.5   Mean   : 722.2   Mean   : 69.24   Mean   : 361.2  
 3rd Qu.:10.000   3rd Qu.: 3890.5   3rd Qu.:1054.0   3rd Qu.: 92.50   3rd Qu.: 497.5  
 Max.   :24.000   Max.   :14053.0   Max.   :4256.0   Max.   :548.00   Max.   :2165.0  
      CRBI            CWalks       League  Division    PutOuts          Assists          Errors      
 Min.   :   3.0   Min.   :   1.0   A:139   E:129    Min.   :   0.0   Min.   :  0.0   Min.   : 0.000  
 1st Qu.:  95.0   1st Qu.:  71.0   N:124   W:134    1st Qu.: 113.5   1st Qu.:  8.0   1st Qu.: 3.000  
 Median : 230.0   Median : 174.0                    Median : 224.0   Median : 45.0   Median : 7.000  
 Mean   : 330.4   Mean   : 260.3                    Mean   : 290.7   Mean   :118.8   Mean   : 8.593  
 3rd Qu.: 424.5   3rd Qu.: 328.5                    3rd Qu.: 322.5   3rd Qu.:192.0   3rd Qu.:13.000  
 Max.   :1659.0   Max.   :1566.0                    Max.   :1377.0   Max.   :492.0   Max.   :32.000  
     Salary       NewLeague
 Min.   :1.438   A:141    
 1st Qu.:1.658   N:122    
 Median :1.800            
 Mean   :1.768            
 3rd Qu.:1.890            
 Max.   :2.055            
>
```

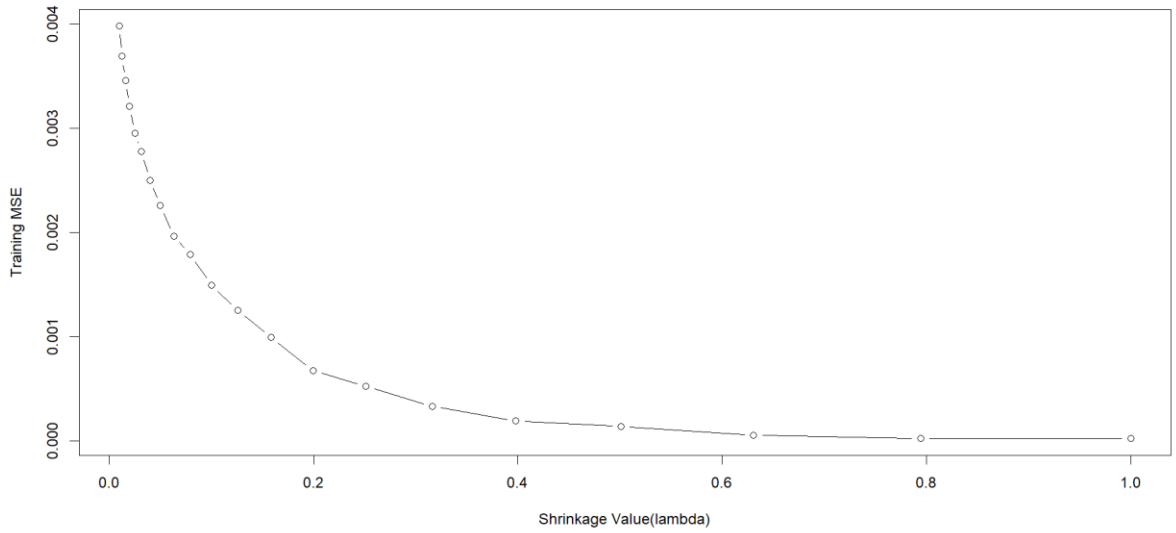
(b) Create a training set consisting of the first 200 observations, and a test set consisting of the remaining observations.

```
> #b)
> subset<-1:200
> hitters.train<-Hitters[subset,]
> hitters.test<-Hitters[-subset,]
> 
```

(c) Perform boosting on the training set with 1,000 trees for a range of values of the shrinkage parameter λ. Produce a plot with different shrinkage values on the x-axis and the corresponding training set MSE on the y-axis.

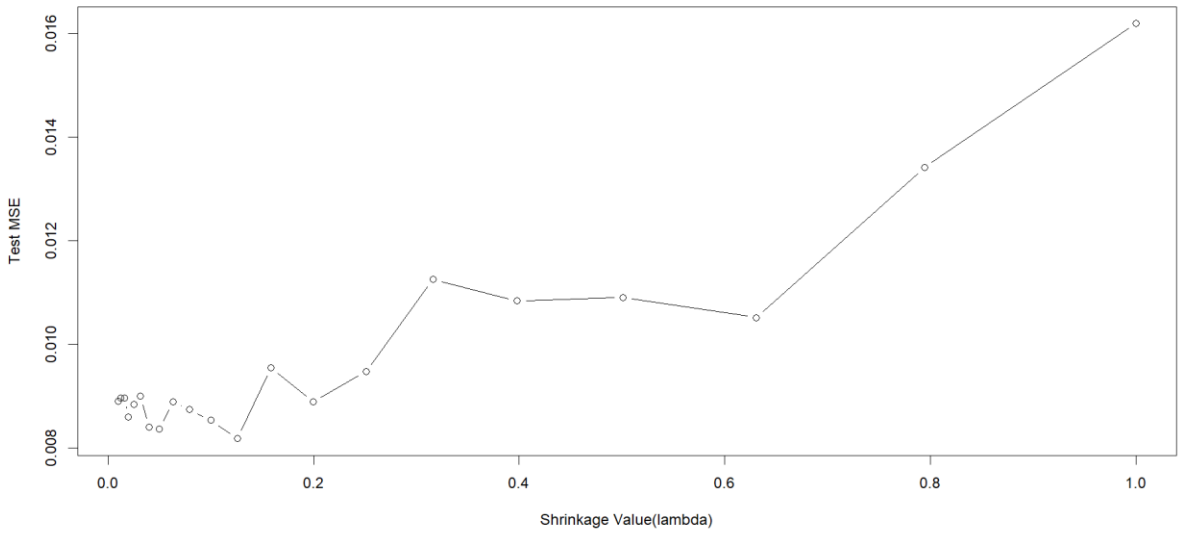
```
> #c)
> install.packages("gbm")
WARNING: Rtools is required to build R packages but is not currently installed. Please download and instal
l the appropriate version of Rtools before proceeding:

https://cran.rstudio.com/bin/windows/Rtools/
Installing package into 'C:/Users/pasun/AppData/Local/R/win-library/4.3'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.3/gbm_2.1.8.1.zip'
Content type 'application/zip' length 894925 bytes (873 KB)
downloaded 873 KB

package 'gbm' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\pasun\AppData\Local\Temp\RtmpGAXddF\downloaded_packages
> library(gbm)
Loaded gbm 2.1.8.1
Warning message:
package 'gbm' was built under R version 4.3.1
```

```
> set.seed(1)
> #defining range of lambdas from 0.01 till 1
> powerss<-seq(-2,0,by=0.1)
> lambdas<-10^powerss
> #defining list storing training errors
> train.error<-rep(NA,length(lambdas))
> #For loop over the range of values of lambdas to store error
> #tuning parameters of boosting:Shrinkage(lambda),number of trees(ntree), & distribution(gaussian for reg
ression trees and binomial for classification trees)
> for (i in 1:length(lambdas)){
+   hitters.gbm<-gbm(Salary~.,hitters.train,distribution = "gaussian",n.trees=1000,shrinkage=lambdas[i])
+
+   #predicting trainig error
+
+   hitters.train.pred<-predict(hitters.gbm,hitters.train,n.trees=1000)
+   train.error[i]<-mean((hitters.train.pred-hitters.train$Salary)^2)
+ }
> #Plotting training MSE against Lambdas
> plot(lambdas,train.error,type="b",xlab="Shrinkage Value(lambda)",ylab="Training MSE")
> 
```



(d) Produce a plot with different shrinkage values on the x-axis and the corresponding test set MSE on the y-axis.

```
> #d)
> #test mse
> set.seed(1)
> #defining list storing testing errors
> test.error<-rep(NA,length(lambdas))
> #For loop over the range of values of lambdas to store error
> #tuning parameters of boosting:Shrinkage(lambda),number of trees(ntree), & distribution(gaussian for reg
ression trees and binomial for classification trees)
> for (i in 1:length(lambdas)){
+   hitters.gbm<-gbm(Salary~.,hitters.train,distribution = "gaussian",n.trees=1000,shrinkage=lambdas[i])
+
+   #predicting testig error
+
+   hitters.test.pred<-predict(hitters.gbm,hitters.test,n.trees=1000)
+   test.error[i]<-mean((hitters.test.pred-hitters.test$Salary)^2)
+ }
> #Plotting testing MSE against Lambdas
> plot(lambdas,test.error,type="b",xlab="Shrinkage Value(lambda)",ylab="Test MSE")
>
```



```
> hitters.gbm.testerror<-min(test.error)
> hitters.gbm.testerror
[1] 0.008174809
>
```

The Minimum Test MSE obtained by boosting is 0.008.

(e) Compare the test MSE of boosting to the test MSE that results from applying two of the regression approaches seen in Chapters 3 and 6.

```
> #e)
> install.packages("glmnet")
WARNING: Rtools is required to build R packages but is not currently installed. Please download and instal
l the appropriate version of Rtools before proceeding:

https://cran.rstudio.com/bin/windows/Rtools/
Installing package into 'C:/Users/pasun/AppData/Local/R/win-library/4.3'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.3/glmnet_4.1-7.zip'
Content type 'application/zip' length 2470450 bytes (2.4 MB)
downloaded 2.4 MB

package 'glmnet' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\pasun\AppData\Local\Temp\RtmpGAXddF\downloaded_packages
> library(glmnet)
Loaded glmnet 4.1-7
Warning message:
package 'glmnet' was built under R version 4.3.1
```

```
> #Fitting least square regression model
> lm<-lm(Salary~.,hitters.train)
> hitters.predict.lm<-predict(lm,hitters.test)
> hitters.lm.test.mse<-mean((hitters.predict.lm-hitters.test$Salary)^2)
> hitters.lm.test.mse
[1] 0.01496996
>
```

Ridge Regression Model:

```
> x<-model.matrix(Salary~.,hitters.train)
> x.test<-model.matrix(Salary ~ . , hitters.test)
> y<-hitters.train$Salary
> hitters.ridge<-glmnet(x,y,alpha=0)
> hitters.ridge.predict<-predict(hitters.ridge,s=0.01,x.test)
> hitters.ridge.test.mse<-mean((hitters.ridge.predict-hitters.test$Salary)^2)
> hitters.ridge.test.mse
[1] 0.01390601
```

Lasso Regression Model;

```
> x<-model.matrix(Salary~.,hitters.train)
> x.test<-model.matrix(Salary ~ . , hitters.test)
> y<-hitters.train$Salary
> hitters.lasso<-glmnet(x,y,alpha=1)
> hitters.lasso.predict<-predict(hitters.lasso,s=0.01,x.test)
> hitters.lasso.test.mse
[1] 0.4700537
```

We have Test MSE for different methods as summarized below. It can be seen Boosting gives least Test MSE among the 4 models

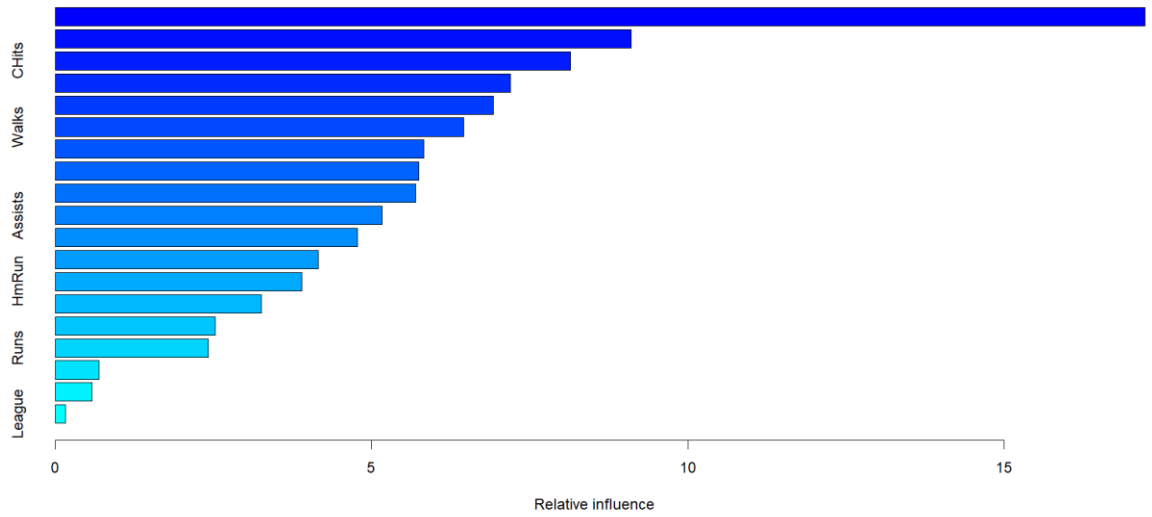
Least Square Regression Full Model Test MSE: 0.14

Ridge Regression Model Test MSE: 0.13

Lasso Regression Model Test MSE: 0.47

(f) Which variables appear to be the most important predictors in the boosted model?

```
> #f)
> boost.hitters<-gbm(Salary~.,data=hitters.train,distribution = "gaussian",n.trees = 1000,shrinkage=lambda
s[which.min(test.error)])
> summary(boost.hitters)
                var    rel.inf
CAtBat       CAtBat 17.2288343
CRuns         CRuns  9.1091223
CHits         CHits  8.1450565
CRBI           CRBI  7.2011839
PutOuts     PutOuts  6.9272967
Walks         Walks  6.4550930
CHmRun       CHmRun  5.8301723
CWalks       CWalks  5.7484755
Years         Years  5.6977407
Assists     Assists  5.1672747
RBI             RBI  4.7822507
Hits           Hits  4.1563824
HmRun         HmRun  3.8941267
AtBat         AtBat  3.2619721
Errors       Errors  2.5319862
Runs           Runs  2.4216832
NewLeague NewLeague  0.6937654
Division   Division  0.5838679
League       League  0.1637156
>
```



We find that CAtbat is the most important variable.

(g) Now apply bagging to the training set. What is the test set MSE for this approach?

```
> #g)
> set.seed(1)
> hitters.bagging<-randomForest(Salary~.,hitters.train,mtry=19,importance=TRUE)
> hitters.bagg.predict<-predict(hitters.bagging,hitters.test)
> hitters.bagg.test.mse<-mean((hitters.bagg.predict-hitters.test$Salary)^2)
> hitters.bagg.test.mse
[1] 0.007282488
>
```

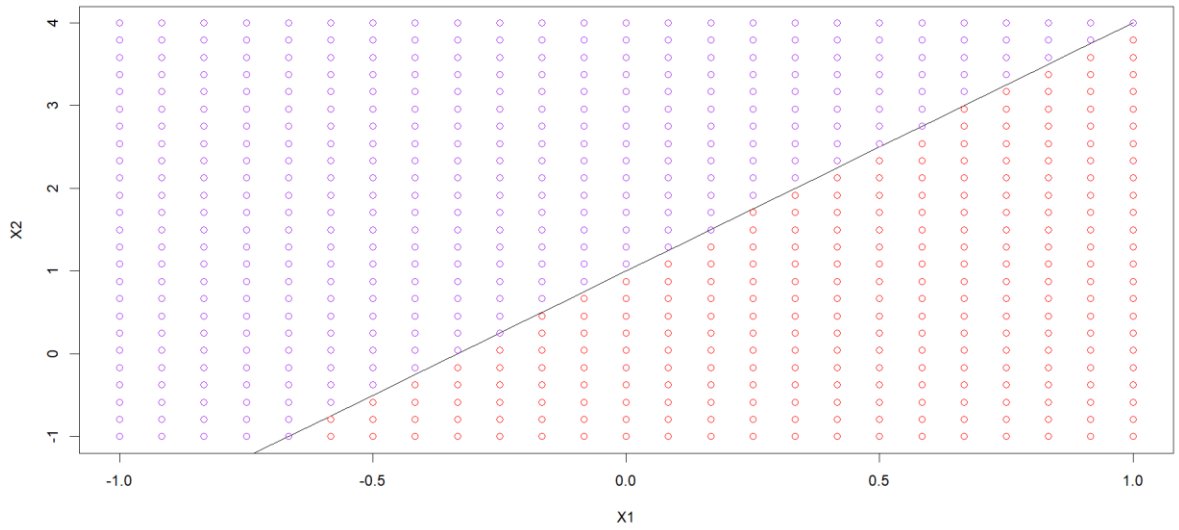
The Test set MSE for Bagging is 0.007.

Chapter 9 Conceptual Problem 1:

This problem involves hyperplanes in two dimensions.

(a) Sketch the hyperplane 1 + 3X1 − X2 = 0. Indicate the set of points for which 1 + 3X1 − X2 > 0, as well as the set of points for which 1 + 3X1 − X2 < 0.

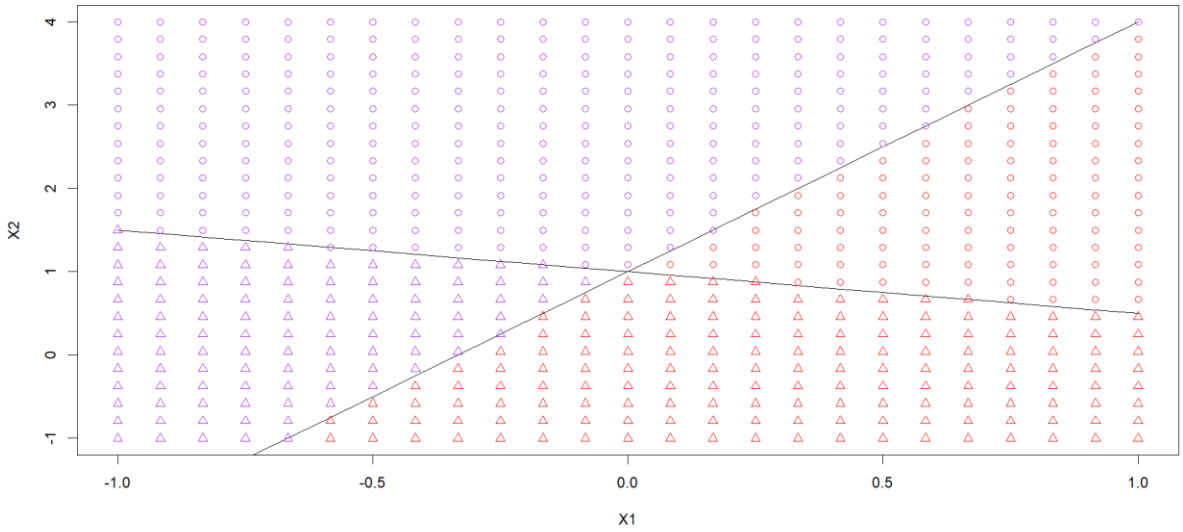
```
> #a)
> X1=seq(-1,1,0.1)
> plot(X1,1+3*X1,xlab='X1',ylab='X2',type='l',xlim=c(-1,1),ylim=c(-1,4))
> for(i in seq(-1,1,length.out = 25)){
+   pts=data.frame(rep(i,25),seq(-1,4,length.out = 25))
+   points(pts,col=ifelse(1+3*pts[,1]-pts[,2]>0,'red','purple'))
+ }
>
```



The plot shows the linear decision boundary for the equation 1 + 3X1 − X2. Purple indicates the values such that they 1 + 3X1 − X2≤0.

(b) On the same plot, sketch the hyperplane −2 + X1 + 2X2 = 0. Indicate the set of points for which −2 + X1 + 2X2 > 0, as well as the set of points for which −2 + X1 + 2X2 < 0.

```
> #b)
> X1=seq(-1,1,0.1)
> plot(X1,1+3*X1,xlab='X1',ylab='X2',type='l',xlim=c(-1,1),ylim=c(-1,4))
> lines(X1,1-1/2*X1)
> for(i in seq(-1,1,length.out = 25)){
+   pts=data.frame(rep(i,25),seq(-1,4,length.out = 25))
+   points(pts,col=ifelse(1+3*pts[,1]-pts[,2]>0,'red','purple'),pch=ifelse(-2+pts[,1]+2*pts[,2]>0,1,2))
+ }
>
```



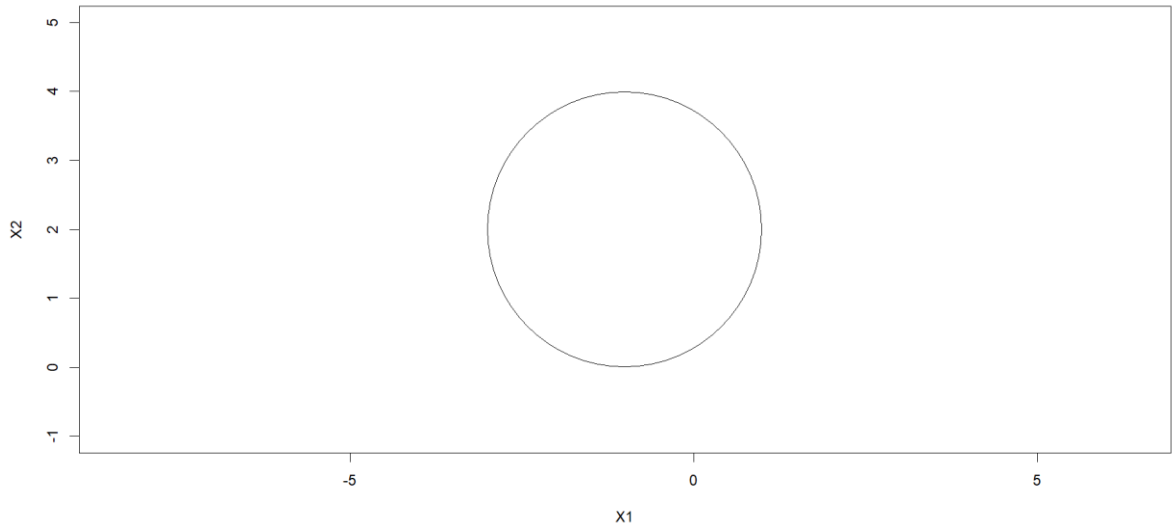
Triangles indicates the points that lie below the second hyper plane.

Chapter 9 Conceptual Problem 2:

We have seen that in p = 2 dimensions, a linear decision boundary takes the form β0+β1X1+β2X2 = 0. We now investigate a non-linear decision boundary.

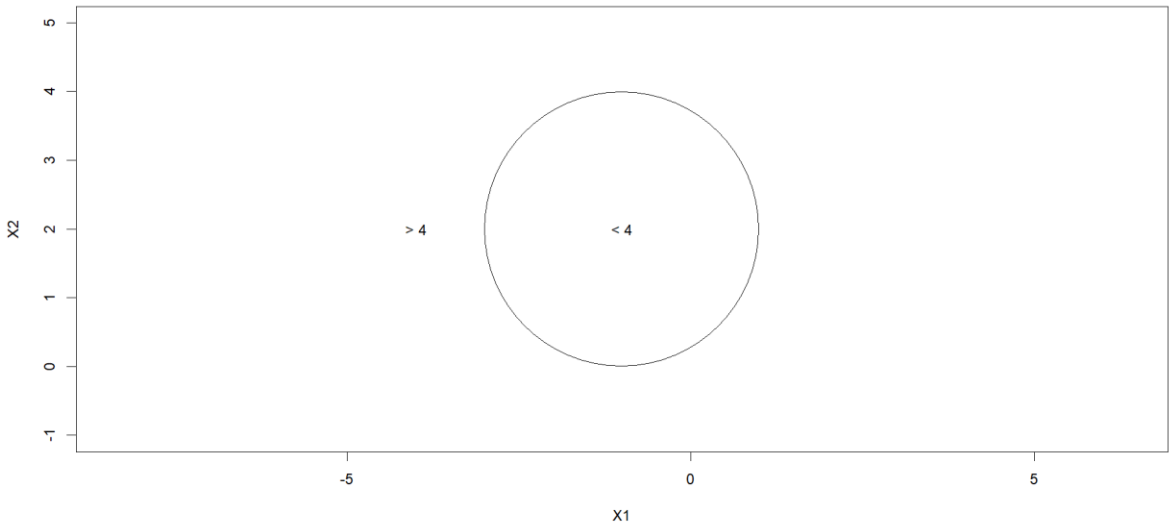
(a) Sketch the curve (1 + X1) 2 + (2 − X2) 2 = 4.

```
> #Chapter 9 Conceptual Problem 2:
> #a)
> plot(NA, NA, type = "n", xlim = c(-4, 2), ylim = c(-1, 5), asp = 1, xlab = "X1", ylab = "X2")
> symbols(c(-1), c(2), circles = c(2), add = TRUE, inches = FALSE)
>
```



(b) On your sketch, indicate the set of points for which (1 + X1) 2 + (2 − X2) 2 > 4, as well as the set of points for which (1 + X1) 2 + (2 − X2) 2 ≤ 4.

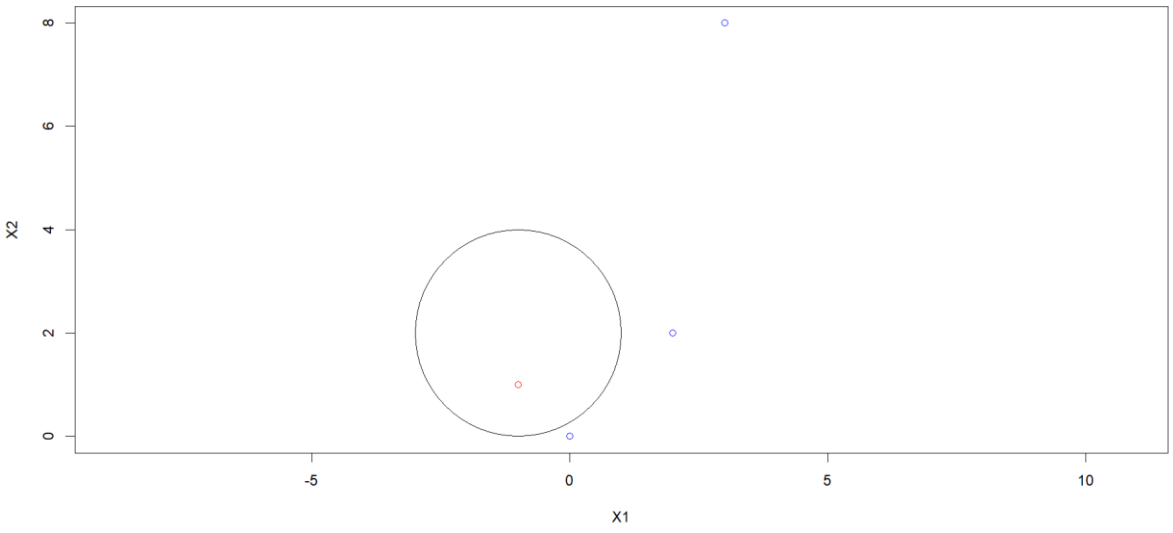
```
> #b)
> plot(NA, NA, type = "n", xlim = c(-4, 2), ylim = c(-1, 5), asp = 1, xlab = "X1", ylab = "X2")
> symbols(c(-1), c(2), circles = c(2), add = TRUE, inches = FALSE)
> text(c(-1), c(2), "< 4")
> text(c(-4), c(2), "> 4")
>
```



(c) Suppose that a classifier assigns an observation to the blue class if (1 + X1) 2 + (2 − X2) 2 > 4, and to the red class otherwise. To what class is the observation (0, 0) classified? (−1, 1)? (2, 2)? (3, 8)?

It is sufficient to substitute the equation's points' coordinates for X_1 and X_2, then check to see if the result is less than or equal to 4. We have 5>4 (blue class) for (0,0), 14 (red class), 9>4 (blue class), 52>4 (blue class), and 14 (red class) for (1,1), (2,2), and (3,8).

```
> plot(c(0, -1, 2, 3), c(0, 1, 2, 8), col = c("blue", "red", "blue", "blue"),
+      type = "p", asp = 1, xlab = "X1", ylab = "X2")
> symbols(c(-1), c(2), circles = c(2), add = TRUE, inches = FALSE)
```



(d) Argue that while the decision boundary in (c) is not linear in terms of X1 and X2, it is linear in terms of X1, X2 1 , X2, and X2 2 .

It is obvious since we may expand the equation of the decision boundary,

$(1 + X_1)^2 + (2 - X_2)^2 = 4$

by

$X_1^2 + X_2^2 + 2X_1 - 4X_2 + 1 = 0$

Which is linear in terms of X_1, X_1^2, X_2 and x_2^2.

Chapter 9 Applied Problem 5:

We have seen that we can fit an SVM with a non-linear kernel in order to perform classification using a non-linear decision boundary. We will now see that we can also obtain a non-linear decision boundary by performing logistic regression using non-linear transformations of the features. 400 9. Support Vector Machines

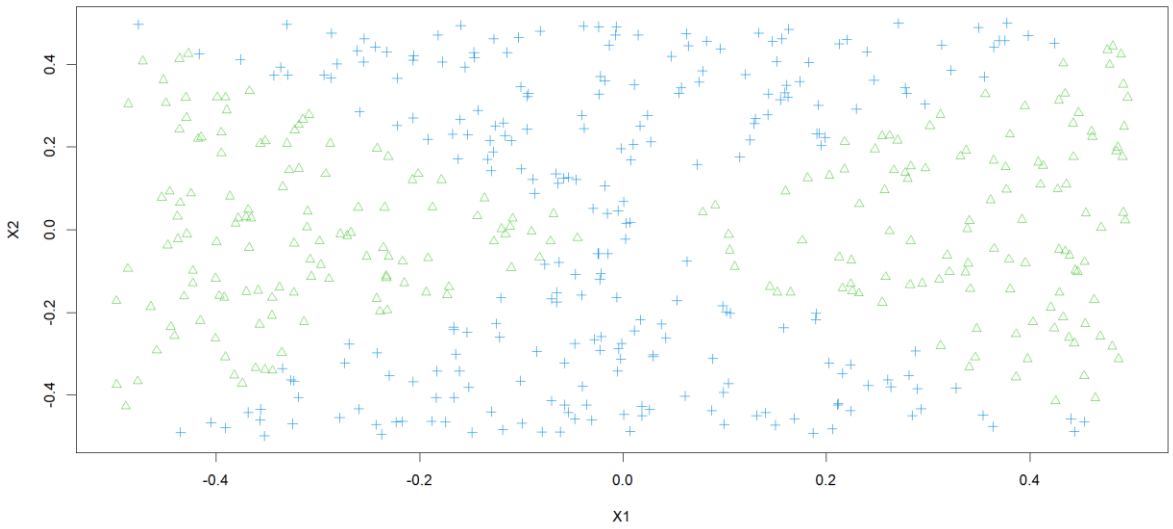
(a) Generate a data set with n = 500 and p = 2, such that the observations belong to two classes with a quadratic decision boundary between them. For instance, you can do this as follows: > x1 <- runif (500) - 0.5 > x2 <- runif (500) - 0.5 > y <- 1 * (x1^2 - x2^2 > 0)

```
> install.packages("ISLR")
WARNING: Rtools is required to build R packages but is not currently installed. Please download and instal
l the appropriate version of Rtools before proceeding:

https://cran.rstudio.com/bin/windows/Rtools/
Installing package into 'C:/Users/pasun/AppData/Local/R/win-library/4.3'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.3/ISLR_1.4.zip'
Content type 'application/zip' length 2924200 bytes (2.8 MB)
downloaded 2.8 MB

package 'ISLR' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\pasun\AppData\Local\Temp\Rtmpw9OL5X\downloaded_packages
> library(ISLR)

Attaching package: 'ISLR'

The following objects are masked _by_ '.GlobalEnv':

    Auto, Hitters

Warning message:
package 'ISLR' was built under R version 4.3.1
>
```

```
> set.seed(1)
> x1 <- runif(500) - 0.5
> x2 <- runif(500) - 0.5
> y <- 1 * (x1^2 - x2^2 > 0)
>
```

(b) Plot the observations, colored according to their class labels. Your plot should display X1 on the x-axis, and X2 on the yaxis.

```
> #b)
> plot(x1, x2, xlab = "X1", ylab = "X2", col = (4 - y), pch = (3 - y))
>
```



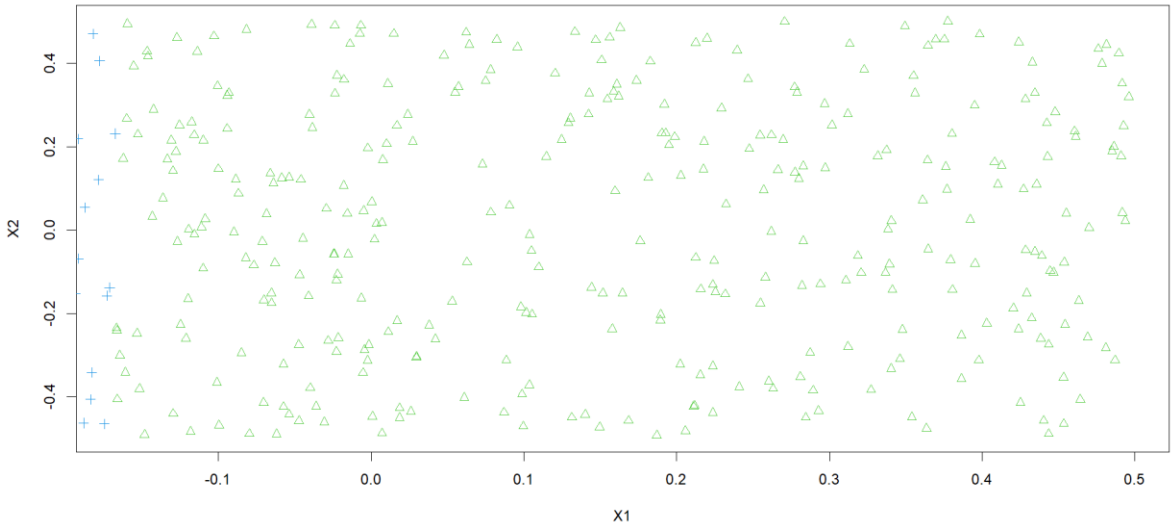
(c) Fit a logistic regression model to the data, using X1 and X2 as predictors.

```
> #c)
> logit.fit <- glm(y ~ x1 + x2, family = "binomial")
> summary(logit.fit)

Call:
glm(formula = y ~ x1 + x2, family = "binomial")

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.087260   0.089579  -0.974    0.330
x1           0.196199   0.316864   0.619    0.536
x2          -0.002854   0.305712  -0.009    0.993

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 692.18  on 499  degrees of freedom
Residual deviance: 691.79  on 497  degrees of freedom
AIC: 697.79

Number of Fisher Scoring iterations: 3

>
```

None of the variables are statistically significants.

(d) Apply this model to the training data in order to obtain a predicted class label for each training observation. Plot the observations, colored according to the predicted class labels. The decision boundary should be linear.

```
> #d)
> data <- data.frame(x1 = x1, x2 = x2, y = y)
> probs <- predict(logit.fit, data, type = "response")
> preds <- rep(0, 500)
> preds[probs > 0.47] <- 1
> plot(data[preds == 1, ]$x1, data[preds == 1, ]$x2, col = (4 - 1), pch = (3 - 1), xlab = "X1", ylab = "X
2")
> points(data[preds == 0, ]$x1, data[preds == 0, ]$x2, col = (4 - 0), pch = (3 - 0))
>
```



The decision boundary is obviously linear.

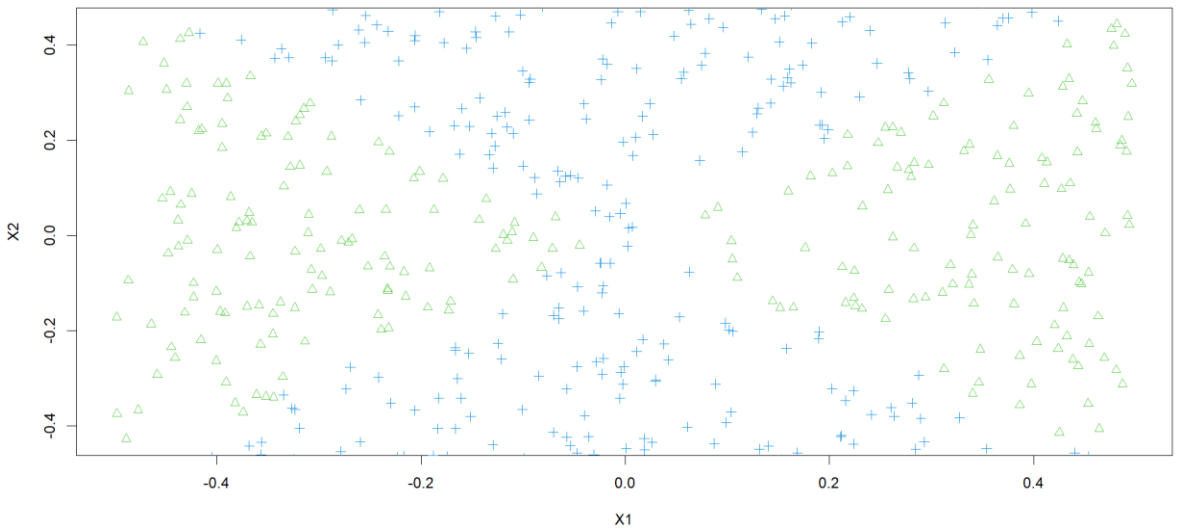
(e) Now fit a logistic regression model to the data using non-linear functions of X1 and X2 as predictors (e.g. X2 1 , X1×X2, log(X2), and so forth).

```
> logitnl.fit <- glm(y ~ poly(x1, 2) + poly(x2, 2) + I(x1 * x2), family = "binomial")
Warning messages:
1: glm.fit: algorithm did not converge
2: glm.fit: fitted probabilities numerically 0 or 1 occurred
> summary(logitnl.fit)

Call:
glm(formula = y ~ poly(x1, 2) + poly(x2, 2) + I(x1 * x2), family = "binomial")

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)    -102.2     4302.0  -0.024    0.981
poly(x1, 2)1   2715.3   141109.5   0.019    0.985
poly(x1, 2)2  27218.5   842987.2   0.032    0.974
poly(x2, 2)1   -279.7    97160.4  -0.003    0.998
poly(x2, 2)2 -28693.0   875451.3  -0.033    0.974
I(x1 * x2)     -206.4    41802.8  -0.005    0.996

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 6.9218e+02  on 499  degrees of freedom
Residual deviance: 3.5810e-06  on 494  degrees of freedom
AIC: 12

Number of Fisher Scoring iterations: 25

>
```

Here again, none of the variables are statistically significant.

(f) Apply this model to the training data in order to obtain a predicted class label for each training observation. Plot the observations, colored according to the predicted class labels. The decision boundary should be obviously non-linear. If it is not, then repeat (a)-(e) until you come up with an example in which the predicted class labels are obviously non-linear.

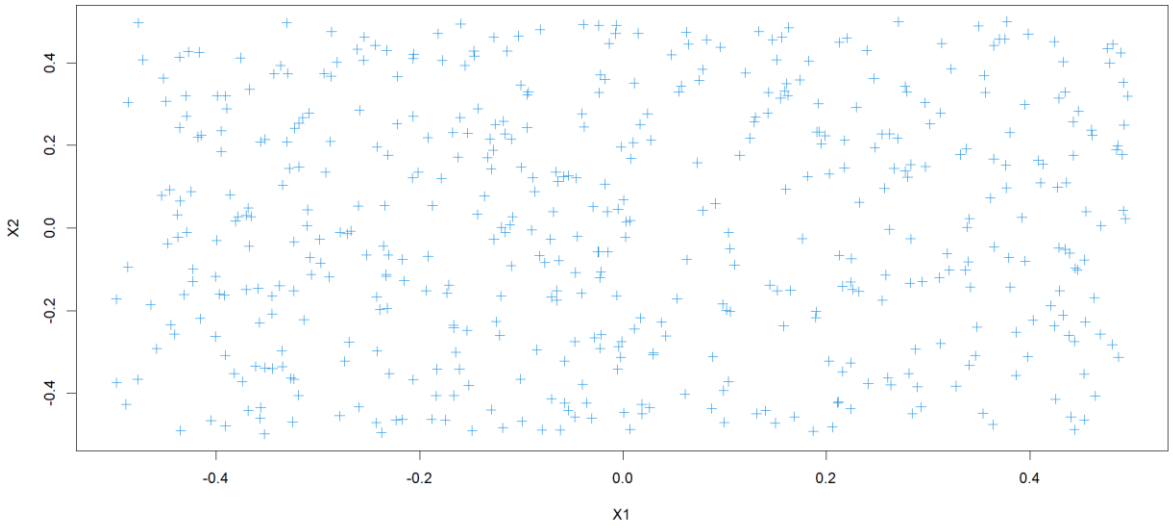
```
> #f)
> probs <- predict(logitnl.fit, data, type = "response")
> preds <- rep(0, 500)
> preds[probs > 0.47] <- 1
> plot(data[preds == 1, ]$x1, data[preds == 1, ]$x2, col = (4 - 1), pch = (3 - 1), xlab = "X1", ylab = "X
2")
> points(data[preds == 0, ]$x1, data[preds == 0, ]$x2, col = (4 - 0), pch = (3 - 0))
>
```



The non-linear decision boundary is surprisingly very similar to the true decision boundary.

(g) Fit a support vector classifier to the data with X1 and X2 as predictors. Obtain a class prediction for each training observation. Plot the observations, colored according to the predicted class labels.

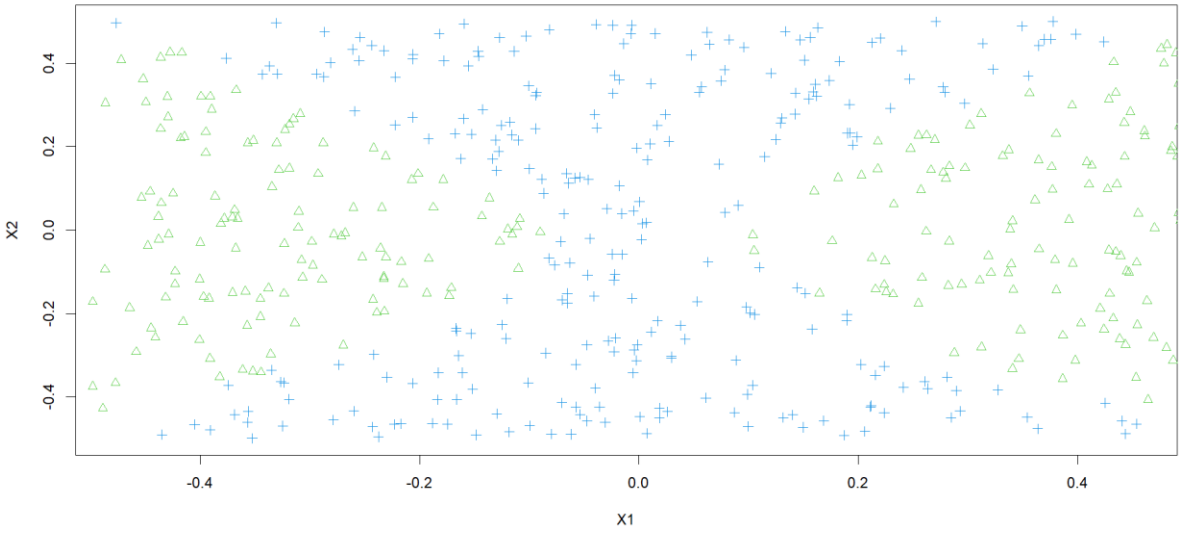
```
> #g)
> install.packages("e1071")
WARNING: Rtools is required to build R packages but is not currently installed. Please download and instal
l the appropriate version of Rtools before proceeding:

https://cran.rstudio.com/bin/windows/Rtools/
Installing package into 'C:/Users/pasun/AppData/Local/R/win-library/4.3'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.3/e1071_1.7-13.zip'
Content type 'application/zip' length 653289 bytes (637 KB)
downloaded 637 KB

package 'e1071' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\pasun\AppData\Local\Temp\Rtmpw90L5X\downloaded_packages
> library(e1071)
Warning message:
package 'e1071' was built under R version 4.3.1
> data$y <- as.factor(data$y)
> svm.fit <- svm(y ~ x1 + x2, data, kernel = "linear", cost = 0.01)
> preds <- predict(svm.fit, data)
> plot(data[preds == 0, ]$x1, data[preds == 0, ]$x2, col = (4 - 0), pch = (3 - 0), xlab = "X1", ylab = "X
2")
> points(data[preds == 1, ]$x1, data[preds == 1, ]$x2, col = (4 - 1), pch = (3 - 1))
>
```



This support vector classifier (even with low cost) classifies all points to a single class.

(h) Fit a SVM using a non-linear kernel to the data. Obtain a class prediction for each training observation. Plot the observations, colored according to the predicted class labels.

```
> #h)
> data$y <- as.factor(data$y)
> svmnl.fit <- svm(y ~ x1 + x2, data, kernel = "radial", gamma = 1)
> preds <- predict(svmnl.fit, data)
> plot(data[preds == 0, ]$x1, data[preds == 0, ]$x2, col = (4 - 0), pch = (3 - 0), xlab = "X1", ylab = "X
2")
> points(data[preds == 1, ]$x1, data[preds == 1, ]$x2, col = (4 - 1), pch = (3 - 1))
>
```



Here again, the non-linear decision boundary is surprisingly very similar to the true decision boundary.

(i) Comment on your results.

We may draw the conclusion that discovering non-linear decision boundaries can be done extremely well using both logistic regression and SVM with non-linear kernels. When it comes to identifying non-linear decision boundaries, SVM with a linear kernel and logistic regression without any interaction terms do very poorly. An argument in favor of SVM, however, is that, in contrast to using SVM, we simply need to tune gamma. This is because when using logistic regression, we must manually tune to identify the appropriate interaction terms.

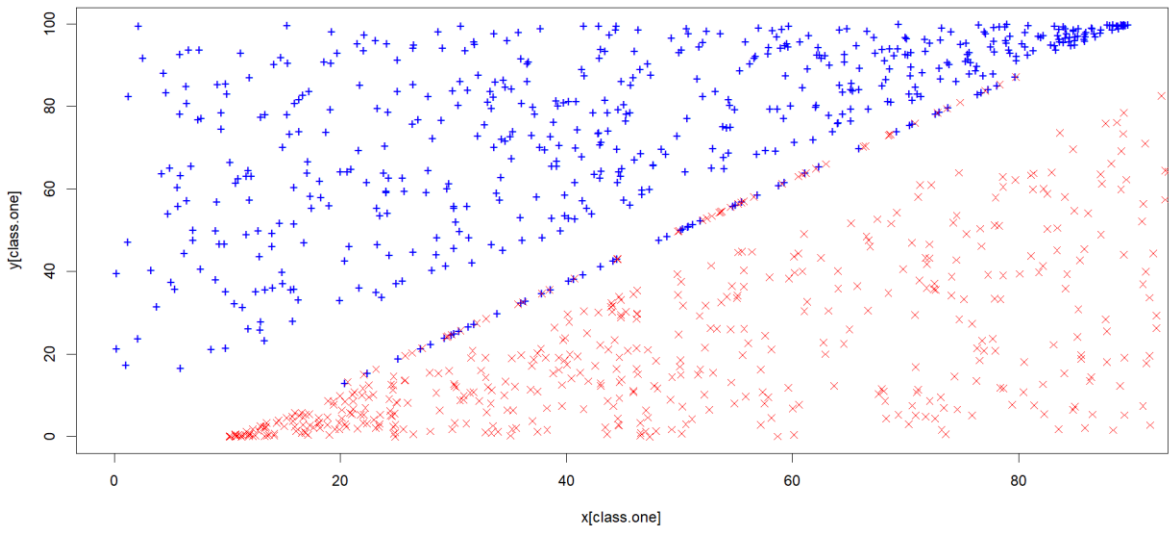
Chapter 9 Applied Problem 6:

At the end of Section 9.6.1, it is claimed that in the case of data that is just barely linearly separable, a support vector classifier with a small value of cost that misclassifies a couple of training observations may perform better on test data than one with a huge value of cost that does not misclassify any training observations. You will now investigate this claim.

(a) Generate two-class data with p = 2 in such a way that the classes are just barely linearly separable.

We randomly generate 1000 points and scatter them across line x=y with wide margin. We also create noisy points along the line 5x−4y−50=0. These points make the classes barely separable and also shift the maximum margin classifier.

```
> set.seed(1)
> x.one <- runif(500, 0, 90)
> y.one <- runif(500, x.one + 10, 100)
> x.one.noise <- runif(50, 20, 80)
> y.one.noise <- 5/4 * (x.one.noise - 10) + 0.1
> x.zero <- runif(500, 10, 100)
> y.zero <- runif(500, 0, x.zero - 10)
> x.zero.noise <- runif(50, 20, 80)
> y.zero.noise <- 5/4 * (x.zero.noise - 10) - 0.1
> class.one <- seq(1, 550)
> x <- c(x.one, x.one.noise, x.zero, x.zero.noise)
> y <- c(y.one, y.one.noise, y.zero, y.zero.noise)
> plot(x[class.one], y[class.one], col = "blue", pch = "+", ylim = c(0, 100))
> points(x[-class.one], y[-class.one], col = "red", pch = 4)
>
```



(b) Compute the cross-validation error rates for support vector classifiers with a range of cost values. How many training errors are misclassified for each value of cost considered, and how does this relate to the cross-validation errors obtained?

```
> install.packages("e1071")
WARNING: Rtools is required to build R packages but is not currently installed. Please download and instal
l the appropriate version of Rtools before proceeding:

https://cran.rstudio.com/bin/windows/Rtools/
Installing package into 'C:/Users/pasun/AppData/Local/R/win-library/4.3'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.3/e1071_1.7-13.zip'
Content type 'application/zip' length 653289 bytes (637 KB)
downloaded 637 KB

package 'e1071' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\pasun\AppData\Local\Temp\RtmpwLI68o\downloaded_packages
> library(e1071)
Warning message:
package 'e1071' was built under R version 4.3.1
```

```
> set.seed(2)
> z <- rep(0, 1100)
> z[class.one] <- 1
> data <- data.frame(x = x, y = y, z = as.factor(z))
> tune.out <- tune(svm, z ~ ., data = data, kernel = "linear", ranges = list(cost = c(0.01, 0.1, 1, 5, 10,
100, 1000, 10000)))

WARNING: reaching max number of iterations
> summary(tune.out)

Parameter tuning of 'svm':

- sampling method: 10-fold cross validation

- best parameters:
  cost
 10000

- best performance: 0

- Detailed performance results:
   cost      error dispersion
1 1e-02 0.06181818 0.03201239
2 1e-01 0.04818182 0.02101641
3 1e+00 0.04818182 0.02101641
4 5e+00 0.05000000 0.02153435
5 1e+01 0.05000000 0.02153435
6 1e+02 0.05272727 0.02259558
7 1e+03 0.04636364 0.03467016
8 1e+04 0.00000000 0.00000000
```

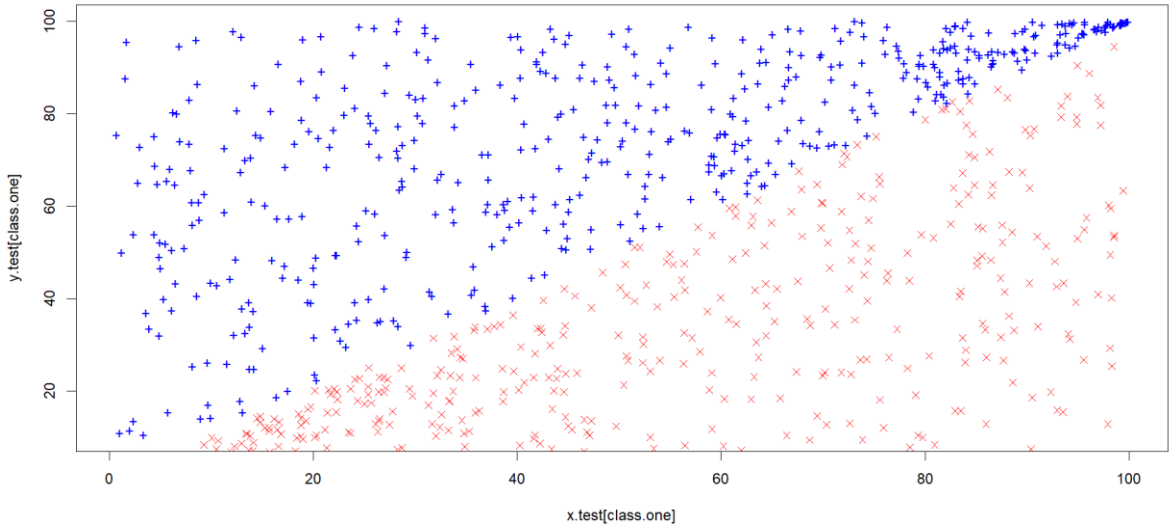
A cost of 10000 seems the best choice of parameter.

```
> data.frame(cost = tune.out$performance$cost, misclass = tune.out$performance$error * 1100)
   cost misclass
1 1e-02       68
2 1e-01       53
3 1e+00       53
4 5e+00       55
5 1e+01       55
6 1e+02       58
7 1e+03       51
8 1e+04        0
>
```

Here a cost of 10000 classify all training points correctly.

(c) Generate an appropriate test data set, and compute the test errors corresponding to each of the values of cost considered. Which value of cost leads to the fewest test errors, and how does this compare to the values of cost that yield the fewest training errors and the fewest cross-validation errors?

```
> #c)
> x.test <- runif(1000, 0, 100)
> class.one <- sample(1000, 500)
> y.test <- rep(NA, 1000)
> # Set y > x for class.one
> for (i in class.one) {
+   y.test[i] <- runif(1, x.test[i], 100)
+ }
> # set y < x for class.zero
> for (i in setdiff(1:1000, class.one)) {
+   y.test[i] <- runif(1, 0, x.test[i])
+ }
> plot(x.test[class.one], y.test[class.one], col = "blue", pch = "+")
> points(x.test[-class.one], y.test[-class.one], col = "red", pch = 4)
>
```



```
> set.seed(3)
> z.test <- rep(0, 1000)
> z.test[class.one] <- 1
> data.test <- data.frame(x = x.test, y = y.test, z = as.factor(z.test))
> costs <- c(0.01, 0.1, 1, 5, 10, 100, 1000, 10000)
> test.err <- rep(NA, length(costs))
> for (i in 1:length(costs)) {
+   svm.fit <- svm(z ~ ., data = data, kernel = "linear", cost = costs[i])
+   pred <- predict(svm.fit, data.test)
+   test.err[i] <- sum(pred != data.test$z)
+ }
> data.frame(cost = costs, misclass = test.err)
   cost misclass
1 1e-02       61
2 1e-01       20
3 1e+00        2
4 5e+00        0
5 1e+01        0
6 1e+02      191
7 1e+03      209
8 1e+04      212
>
```

Costs of 1, 5 or 10 seem to perform better on test observations, this is much smaller than the value of 10000 for training observations.

(d) Discuss your results.

We see overfitting for linear kernel once more. A big cost overfits the train data by attempting to accurately identify noisy points. However, a modest cost works better on test data and makes a few mistakes on the noisy test points.