Homework 7

Jiao Qu A20386614, Yuan-An Liu A20375099, Zhenyu Zhang A20287371

11/21/2017

R Markdown

1. Create a new column in the dataset, high_mileage that is true if mpg > mean(mpg). Else its false. You will try to predict these variables as a function of the predictors. DO NOT USE name and origin is a categorical as before.

```
library(ISLR)
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 3.4.2
## Loading required package: Matrix
## Loading required package: foreach
## Loaded glmnet 2.0-13
```

```
library(tree)
library(gbm)
```

```
## Loading required package: survival
## Loading required package: lattice
## Loading required package: splines
## Loading required package: parallel
## Loaded gbm 2.1.3
```

```
library(e1071)
data(Auto)
table = Auto[1:7]
meanOfmpg = mean(Auto$mpg)
high_mileage=ifelse(Auto$mpg<meanOfmpg,"false","true")
newTable = data.frame(table,high_mileage)
summary(newTable)
```

```
##       mpg          cylinders      displacement     horsepower
##  Min.   : 9.00   Min.   :3.000   Min.   : 68.0   Min.   : 46.0
##  1st Qu.:17.00   1st Qu.:4.000   1st Qu.:105.0   1st Qu.: 75.0
##  Median :22.75   Median :4.000   Median :151.0   Median : 93.5
##  Mean   :23.45   Mean   :5.472   Mean   :194.4   Mean   :104.5
##  3rd Qu.:29.00   3rd Qu.:8.000   3rd Qu.:275.8   3rd Qu.:126.0
##  Max.   :46.60   Max.   :8.000   Max.   :455.0   Max.   :230.0
##      weight      acceleration        year       high_mileage
##  Min.   :1613   Min.   : 8.00   Min.   :70.00   false:206
##  1st Qu.:2225   1st Qu.:13.78   1st Qu.:73.00   true :186
##  Median :2804   Median :15.50   Median :76.00
##  Mean   :2978   Mean   :15.54   Mean   :75.98
##  3rd Qu.:3615   3rd Qu.:17.02   3rd Qu.:79.00
##  Max.   :5140   Max.   :24.80   Max.   :82.00
```

2. Set the seed to one and set up the data for 3-fold cross validation

```
k=3
set.seed(1)
folds=sample(1:k,nrow(newTable),replace = TRUE)
tableSet = data.frame(newTable,folds)
Auto$origin=factor(Auto$origin)
```

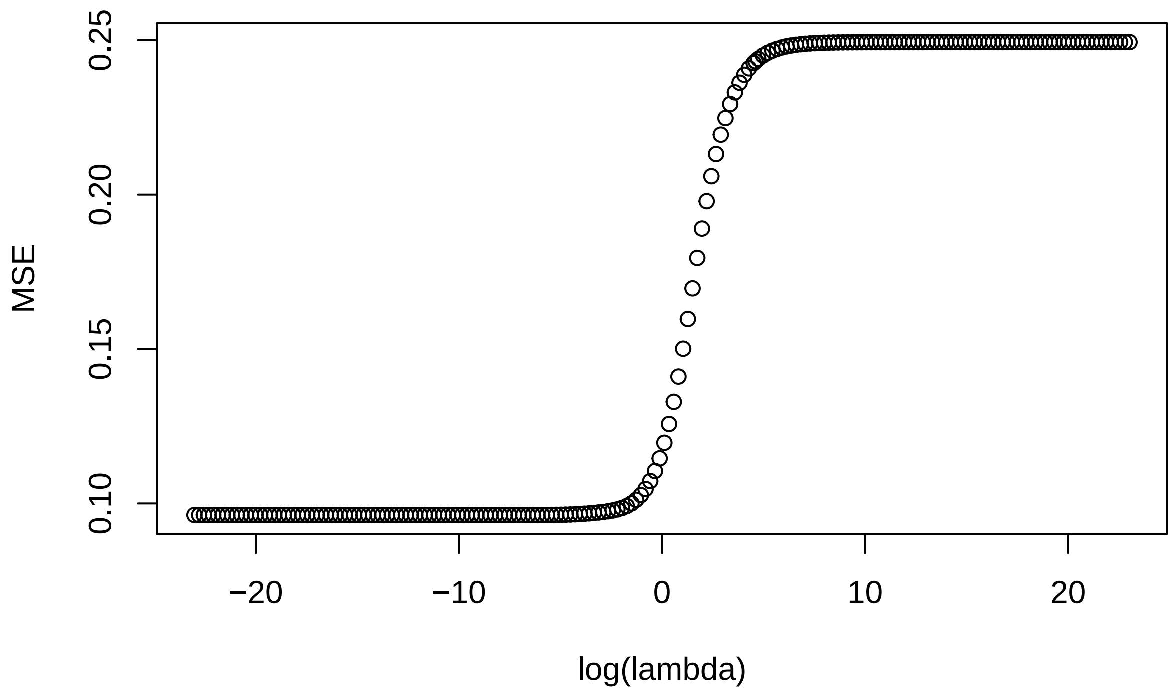
3. Guess which classifier will do best.

Answer:Boosting is best

4. Predict high mpg with these classifiers:

a. Logistic regression tuning (i.e., with ridge regularization)

```
high_mileage <- Auto$mpg > mean(Auto$mpg)
Auto$origin <- as.factor(Auto$origin)

Auto = data.frame(Auto,high_mileage)[,c(-1,-9)]
set.seed(1)
k = 3
folds = sample(1:k, nrow(Auto), rep = TRUE)
x = model.matrix(high_mileage ~ ., Auto)[,-1]
y = Auto$high_mileage
expo = seq(-10, 10, length = 200)
lmd = sort(c(0, 100, 10 ^ expo))

MSErg = c()
for (i in 1:k){
        train = c(1:nrow(x))[folds != i]
        test = (-train)
        y.test = y[test]

      glm.fit = glmnet(x[train,], y[train], alpha = 0, lambda = lmd, thresh = 1e-12)
        glm.row = nrow(x[test,])

        glm.probs = predict(glm.fit, s = lmd, newx = x[test,])


        if (!length(MSErg)){
          MSErg = rep(0, ncol(glm.probs))
          }
        MSErg = MSErg + colMeans((glm.probs - y.test) ^ 2)
}
MSErg = unname(MSErg) / k

plot(x = log(lmd), y = MSErg, xlab = 'log(lambda)', ylab = 'MSE')
```




```r
print("The Lowest MSE:")
```

```
## [1] "The Lowest MSE:"
```

```r
print(min(MSErg))
```

```
## [1] 0.09625485
```

```r
optirg = lmd[which(MSErg == min(MSErg))]
print("The Best Lambda:")
```

```
## [1] "The Best Lambda:"
```

```r
print(optirg)
```

```
## [1] 0.001366716
```

```r
Mrate = 0
for (i in 1:k){
        train = c(1:nrow(x))[folds != i]
        test = (-train)
        y.test = y[test]

      glm.fit = glmnet(x[train,], y[train], alpha = 0, lambda = lmd, thresh = 1e-12)
        glm.row = nrow(x[test,])

        glm.probs = predict(glm.fit, s = optirg, newx = x[test,])

      glm.pred = rep(FALSE,glm.row)
        glm.pred[glm.probs > 0.5] = TRUE
      print(table(glm.pred, y.test))
        print(mean(glm.pred == y.test))
        Mrate = Mrate + mean(glm.pred == y.test)
}
```

```
##         y.test
## glm.pred FALSE TRUE
##    FALSE    59    1
##    TRUE     10   61
## [1] 0.9160305
##         y.test
## glm.pred FALSE TRUE
##    FALSE    57    2
##    TRUE     15   65
## [1] 0.8776978
##         y.test
## glm.pred FALSE TRUE
##    FALSE    51    2
##    TRUE     14   55
## [1] 0.8688525
```

```
print("The mean accuracy:")
```

```
## [1] "The mean accuracy:"
```

```
print(Mrate/k)
```

```
## [1] 0.8875269
```

b. Decision trees with tuning (e.g., you will set the splitting criterion)

```
library(ISLR)
library(glmnet)
library(tree)
library(gbm)
library(e1071)
data(Auto)
table = Auto[1:7]
meanOfmpg = mean(Auto$mpg)
high_mileage=ifelse(Auto$mpg<meanOfmpg,"false","true")
newTable = data.frame(table,high_mileage)
summary(newTable)
```

```
##       mpg          cylinders      displacement     horsepower
##  Min.   : 9.00   Min.   :3.000   Min.   : 68.0   Min.   : 46.0
##  1st Qu.:17.00   1st Qu.:4.000   1st Qu.:105.0   1st Qu.: 75.0
##  Median :22.75   Median :4.000   Median :151.0   Median : 93.5
##  Mean   :23.45   Mean   :5.472   Mean   :194.4   Mean   :104.5
##  3rd Qu.:29.00   3rd Qu.:8.000   3rd Qu.:275.8   3rd Qu.:126.0
##  Max.   :46.60   Max.   :8.000   Max.   :455.0   Max.   :230.0
##      weight      acceleration        year       high_mileage
##  Min.   :1613   Min.   : 8.00   Min.   :70.00   false:206
##  1st Qu.:2225   1st Qu.:13.78   1st Qu.:73.00   true :186
##  Median :2804   Median :15.50   Median :76.00
##  Mean   :2978   Mean   :15.54   Mean   :75.98
##  3rd Qu.:3615   3rd Qu.:17.02   3rd Qu.:79.00
##  Max.   :5140   Max.   :24.80   Max.   :82.00
```

```
Auto$origin=factor(Auto$origin)
table = Auto[1:8]
meanOfmpg = mean(Auto$mpg)
k=3
```

```r
set.seed(1)
folds=sample(1:k,nrow(table),replace = TRUE)
high_mileage =ifelse(Auto$mpg<meanOfmpg,"false","true")
newTable = data.frame(table[2:7],high_mileage)
tableSet = data.frame(newTable,folds)
accuracy = rep(0,3)
for(i in 1:3){
  train = tableSet[which(tableSet$folds != i),]
  test = tableSet[which(tableSet$folds == i),]

  tree.mpg = tree(high_mileage~.,data = train)
  cv.mpg =cv.tree(tree.mpg ,FUN=prune.misclass )
  prune.mpg=prune.misclass(tree.mpg,best=4)
  tree.pred=predict(prune.mpg,test,type="class")
  a=table(tree.pred,test$high_mileage)
  accuracy[i] =  (a[1,1] + a[2,2])/(a[1,1]+a[1,2]+a[2,1]+a[2,2])
}

meanAccuracy = (accuracy[1] + accuracy[2] +accuracy[3])/3
meanAccuracy
```

```
## [1] 0.9057141
```

c. Bagging with tuning

```r
Auto$origin=factor(Auto$origin)

library(randomForest)
```

```
## randomForest 4.6-12

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

```r
accuracy = rep(0,3)

for(i in 1:k){
  train = tableSet[which(tableSet$folds != i),]
  test = tableSet[which(tableSet$folds == i),]
  bag.mpg = randomForest(high_mileage ~. , data= train,mtry=dim(newTable)[2],importance = TRUE)
  bag.predict.mpg = predict(bag.mpg , test, type = "class")
  table(bag.predict.mpg,test$high_mileage)
  a=table(bag.predict.mpg,test$high_mileage)[1,1]
  b=table(bag.predict.mpg,test$high_mileage)[2,2]
  total = table(bag.predict.mpg,test$high_mileage)[1,1] +table(bag.predict.mpg,test$high_mileage)[1,2]+
  accuracy[i] = (a + b)/total

}

meanAccuracy =(accuracy[1]+accuracy[2]+accuracy[3]) /3
meanAccuracy
```

```
## [1] 0.9021671
```

d. Random forest with tuning

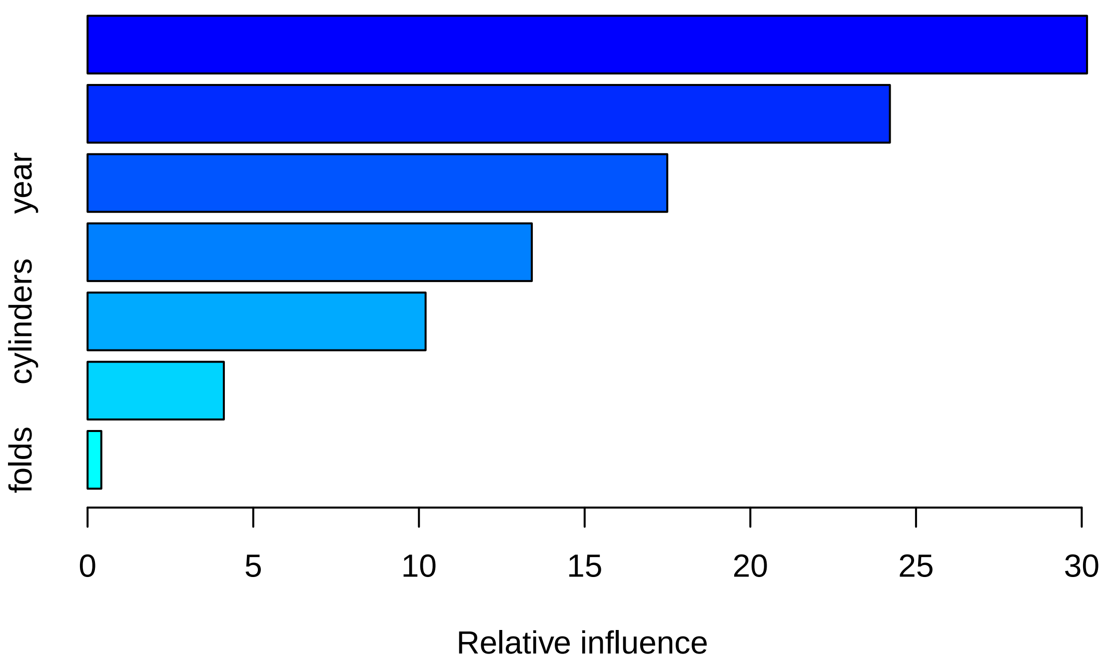
```r
library(randomForest)
Auto$origin=factor(Auto$origin)
```

```
accuracy = rep(0,3)

for(i in 1:k){
  train = tableSet[which(tableSet$folds != i),]
  test = tableSet[which(tableSet$folds == i),]
  random.mpg = randomForest(high_mileage ~. , data= train,mtry=dim(newTable)[2]/3,importance = TRUE)
  random.predict.mpg = predict(random.mpg , test, type = "class")
  table(random.predict.mpg,test$high_mileage)
  a=table(random.predict.mpg,test$high_mileage)[1,1]
  b=table(random.predict.mpg,test$high_mileage)[2,2]
  total = table(random.predict.mpg,test$high_mileage)[1,1] +table(random.predict.mpg,test$high_mileage)
  accuracy[i] = (a + b)/total

}

meanAccuracy =(accuracy[1]+accuracy[2]+accuracy[3]) /3
meanAccuracy
```

```
## [1] 0.9125742
```

e. Boosting with tuning

```
Auto$origin=factor(Auto$origin)
newTable$high_mileage =as.integer(as.logical(newTable$high_mileage))
tableSet = data.frame(newTable,folds)
  train = tableSet[which(tableSet$folds != 1),]
  test = tableSet[which(tableSet$folds == 1),]
boost.mpg = gbm(high_mileage~.,data = train,distribution = "bernoulli",n.trees=1000, interaction.depth=
summary(boost.mpg)
```




```
##                     var    rel.inf
## displacement displacement 30.1604563
## weight             weight 24.2112762
## year                 year 17.4935990
```

```
## horsepower     horsepower 13.4060714
## cylinders       cylinders 10.2009440
## acceleration acceleration  4.1126235
## folds               folds  0.4150296
```

```r
library(ISLR)
library(glmnet)
library(tree)
library(gbm)
library(e1071)
library(survival)
library(pROC)
```

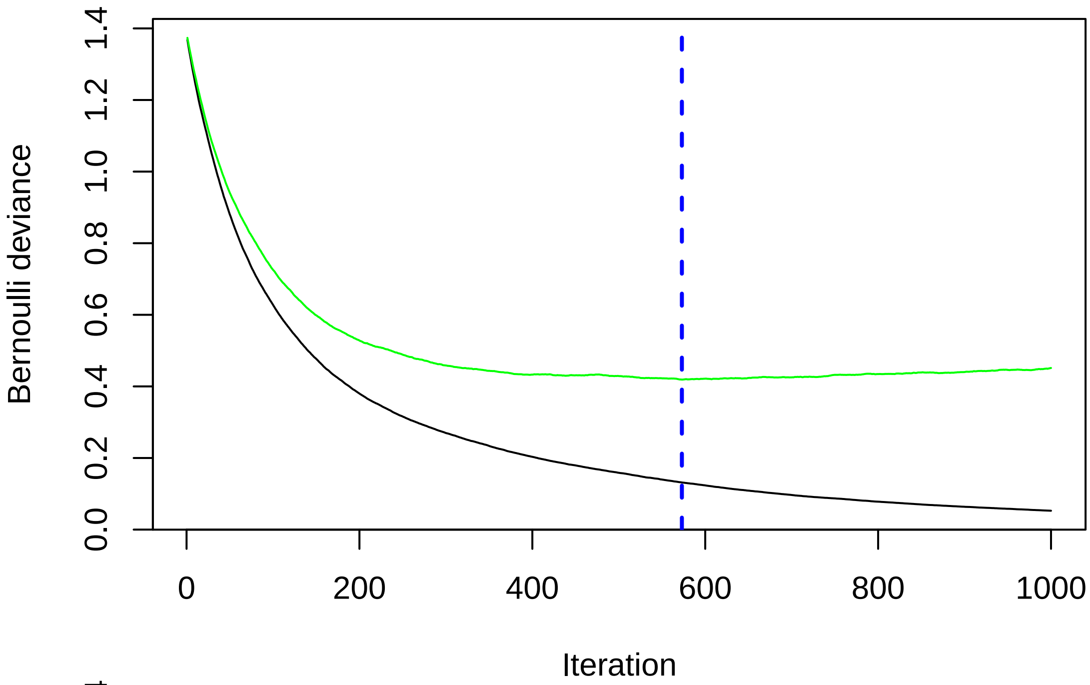
```
## Type 'citation("pROC")' for a citation.
```

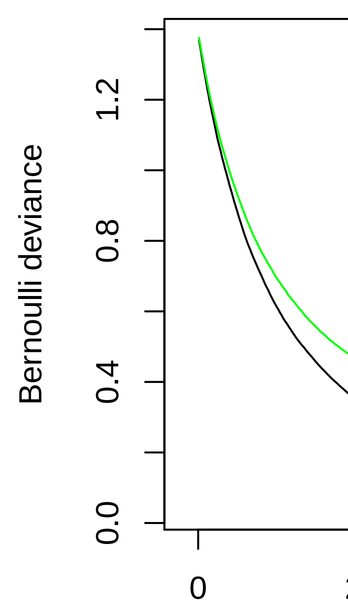
```
##
## Attaching package: 'pROC'
```

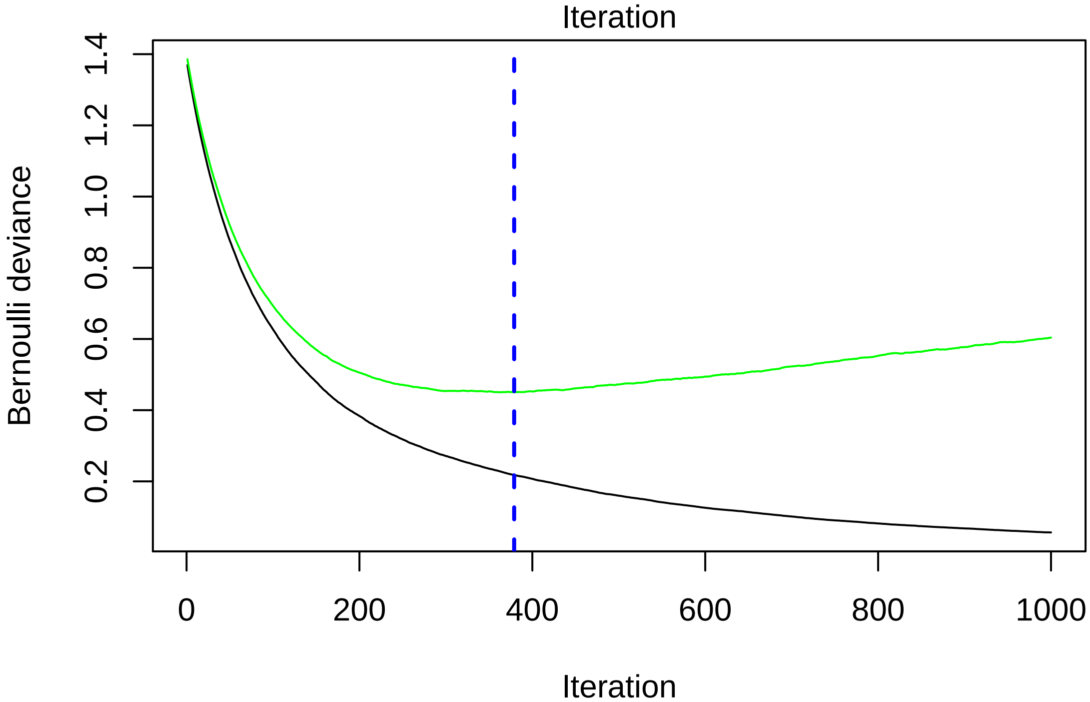
```
## The following object is masked from 'package:glmnet':
##
##     auc
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```r
data(Auto)
Auto$origin=factor(Auto$origin)
table = Auto[1:8]
meanOfmpg = mean(Auto$mpg)
k=3
set.seed(1)
folds=sample(1:k,nrow(table),replace = TRUE)
high_mileage =ifelse(Auto$mpg<meanOfmpg,"false","true")

newTable = data.frame(table[2:7],high_mileage)
tableSet = data.frame(newTable,folds)

tableSet$high_mileage =as.integer(as.logical(tableSet$high_mileage))
library(randomForest)
accuracy = rep(0,3)
for(i in 1:k){
  train = tableSet[which(tableSet$folds != i),]
  test = tableSet[which(tableSet$folds == i),]
  boost.mpg = gbm(high_mileage~.,data = train,distribution = "bernoulli",n.trees=1000, interaction.dept


  boost.iter=gbm.perf(boost.mpg,method = "cv")
  boost.predict = predict(boost.mpg,test,n.trees = boost.iter )
  boost.roc = roc(test$high_mileage,boost.predict)

  coords(boost.roc,"best")
  boost.predict.class = ifelse(boost.predict > coords(boost.roc,"best")["threshold"],"TURE","FALSE")
  a = table(boost.predict.class,test$high_mileage)
  accuracy[i] = (a[1,1] + a[2,2])/(a[1,1]+a[1,2]+a[2,1]+a[2,2])
}
```








```
meanAccuracy = (accuracy[1] + accuracy[2] +accuracy[3])/3
meanAccuracy
```

```
## [1] 0.9209586
```

f. SVM with linear kernel tuning

```
library(ISLR)
library(e1071)
Auto$origin=factor(Auto$origin)
table = Auto[1:8]
meanOfmpg = mean(Auto$mpg)
```

```
k=3
set.seed(1)
folds=sample(1:k,nrow(table),replace = TRUE)
high_mileage =ifelse(Auto$mpg<meanOfmpg,"false","true")
newTable = data.frame(table[2:7],high_mileage)
tableSet = data.frame(newTable,folds)
accuracy = rep(0,3)

for(i in 1:k){
  train = tableSet[which(tableSet$folds != i),]
  test = tableSet[which(tableSet$folds == i),]
  tune.out=tune(svm,high_mileage~.,data=train,kernel="linear",
                ranges=list(cost=c(0.001, 0.01, 0.1, 1,5,10,100)))
  bestmod = tune.out$best.model
  mpg.pred = predict(bestmod,test)
  a = table(predict=mpg.pred,truth = test$high_mileage)
  accuracy[i] = (a[1,1] + a[2,2])/(a[1,1]+a[1,2]+a[2,1]+a[2,2])
}
meanAccuracy = (accuracy[1] + accuracy[2] +accuracy[3])/3
meanAccuracy
```

```
## [1] 0.8922818
```

g. SVM with polynomial kernel and tuning

```
library(ISLR)
library(e1071)
Auto$origin=factor(Auto$origin)
table = Auto[1:8]
meanOfmpg = mean(Auto$mpg)
k=3
set.seed(1)
folds=sample(1:k,nrow(table),replace = TRUE)
high_mileage =ifelse(Auto$mpg<meanOfmpg,"false","true")
newTable = data.frame(table[2:7],high_mileage)
tableSet = data.frame(newTable,folds)
accuracy = rep(0,3)

for(i in 1:k){
  train = tableSet[which(tableSet$folds != i),]
  test = tableSet[which(tableSet$folds == i),]
  tune.out=tune(svm,high_mileage~.,data=train,kernel="polynomial",
                ranges=list(cost=c(0.001, 0.01, 0.1, 1,5,10,100)))
  bestmod = tune.out$best.model
  mpg.pred = predict(bestmod,test)
  a = table(predict=mpg.pred,truth = test$high_mileage)
  accuracy[i] = (a[1,1] + a[2,2])/(a[1,1]+a[1,2]+a[2,1]+a[2,2])
}
meanAccuracy = (accuracy[1] + accuracy[2] +accuracy[3])/3
meanAccuracy
```

```
## [1] 0.8846483
```

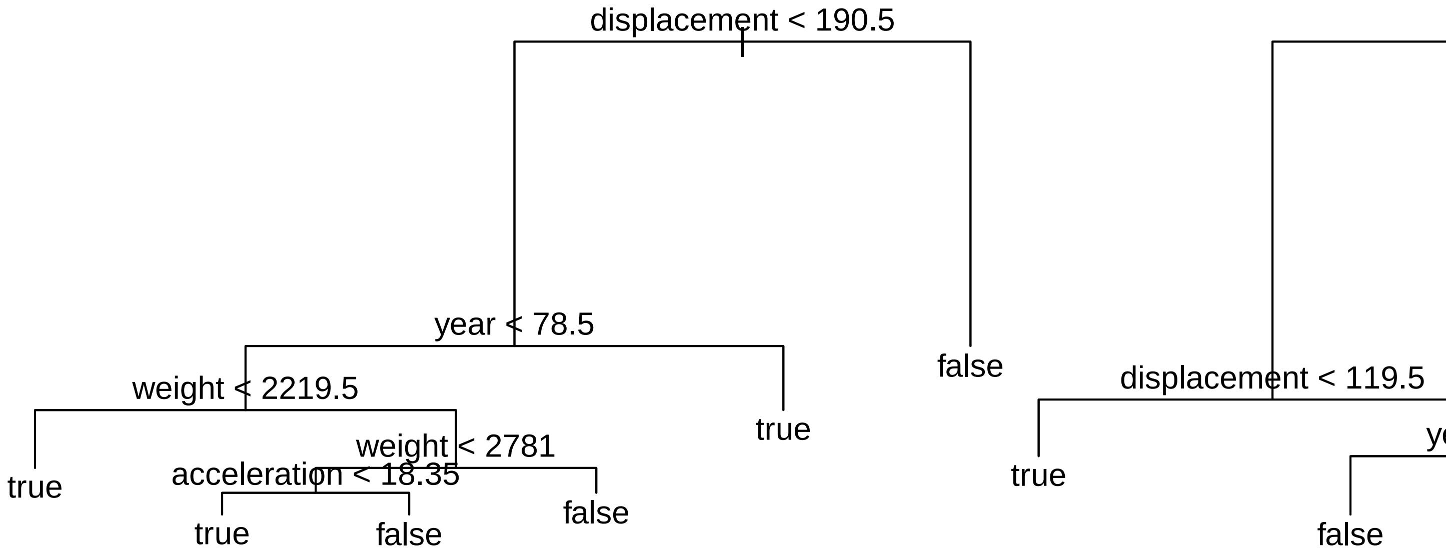
Report a. Report the accuracy if you predicted the most frequent class for all observations. This is your baseline.

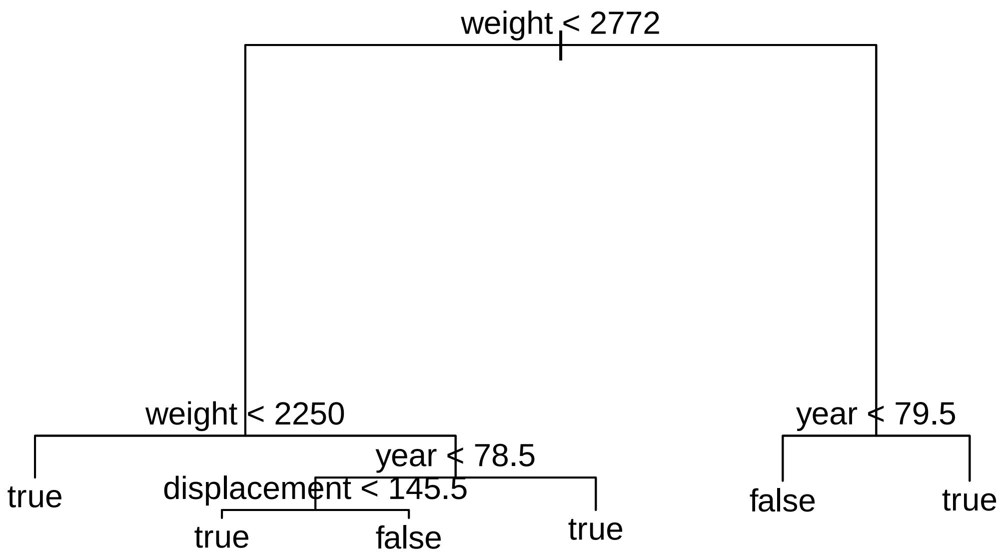
Logistic regression tuning:0.8875269 Decision trees with tuning:0.9057141 Bagging with tuning:0.9048993 Random forest with tuning:0.9180386 Boosting with tuning:0.9209586 SVM with linear kernel tuning:0.8922818 SVM with polynomial kernel and tuning:0.8846483

b. Plot of cross validation accuracy as a function of the tuning parameter for each classifier.

b.1 For the decision tree, plot the tree.

```
library(ISLR)
library(tree)
Auto$origin=factor(Auto$origin)
table = Auto[1:8]
meanOfmpg = mean(Auto$mpg)
k=3
set.seed(1)
folds=sample(1:k,nrow(table),replace = TRUE)
high_mileage =ifelse(Auto$mpg<meanOfmpg,"false","true")
newTable = data.frame(table[2:7],high_mileage)
tableSet = data.frame(newTable,folds)
accuracy = rep(0,3)
for(i in 1:3){
  train = tableSet[which(tableSet$folds != i),]
  test = tableSet[which(tableSet$folds == i),]

  tree.mpg = tree(high_mileage~.,data = train)
  cv.mpg =cv.tree(tree.mpg ,FUN=prune.misclass )
  prune.mpg=prune.misclass(tree.mpg,best=4)
  plot(prune.mpg)
  text(prune.mpg ,pretty=0)
}
```





c. Which classifier does best?

Answer???Boosting with tuning does best since the accuracy 0.9209586 is the highest.

d. Which one would you use? And does this classifier match your initial guess?

Answer : We would use Boosting with tuning. This classifier match our initial guess.