HW7

Anish Mohan

November 16, 2015

# Q1

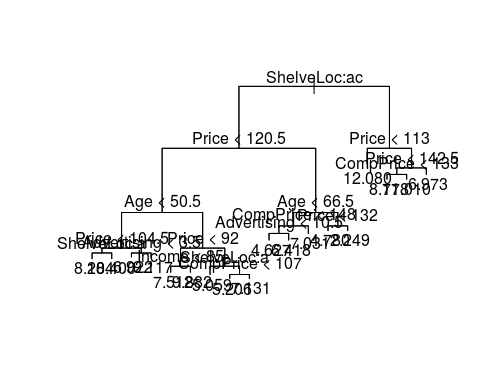
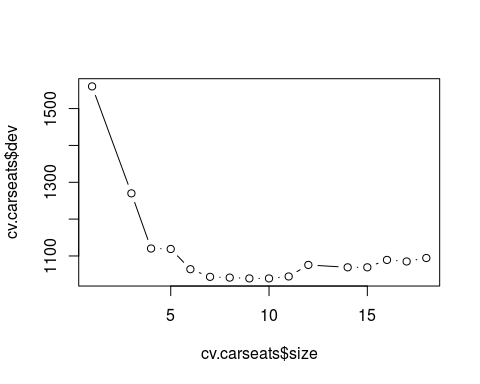
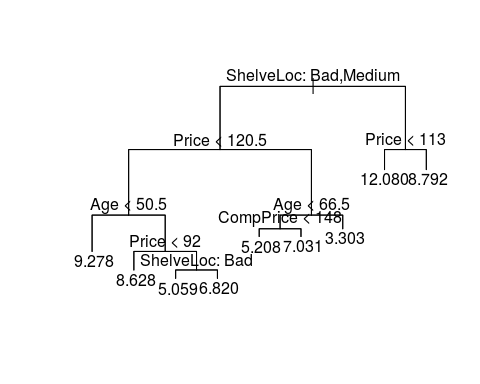
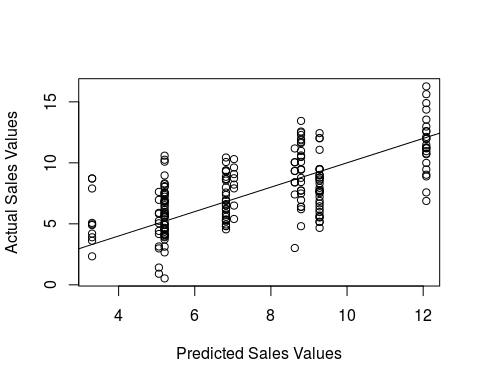
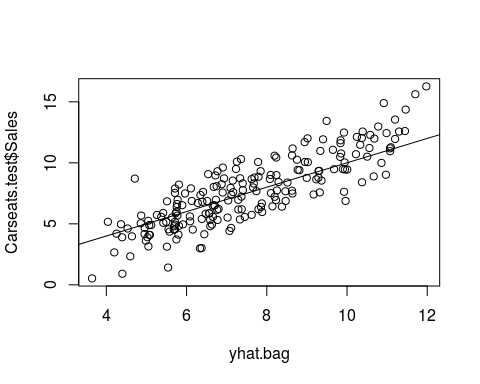
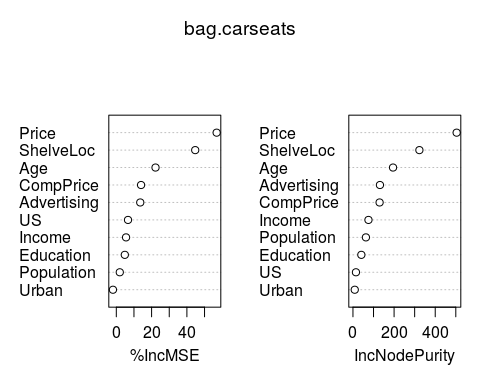
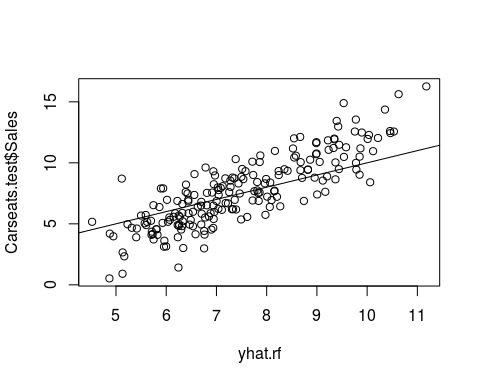
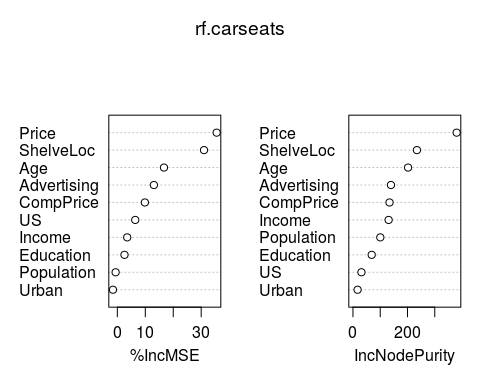
# Q2

* Booststrap estimates for P(Class is Red | X)={ 0.1, 0.15, 0.2, 0.2, 0.55,0.6, 0.6, 0.65, 0.7, 0.75}
* Majority Approach: X=Red

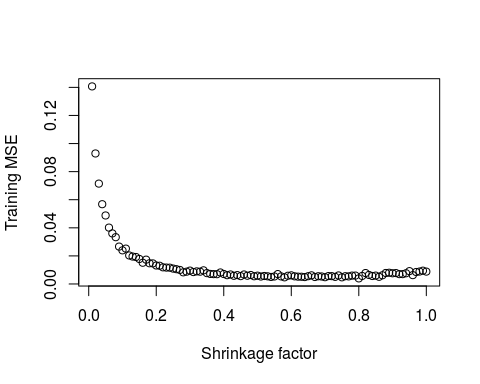
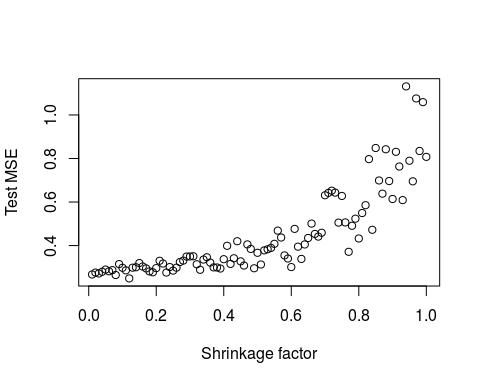
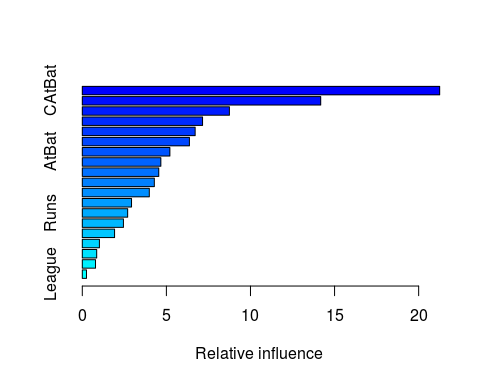
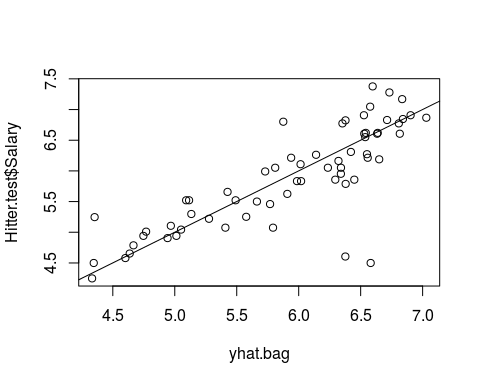
# of time P(Class is Red | X)>0.5. In this case 6 times P(Class is Red | X)>0.5 hence X=Red

* Average Approach: X=Green
* Take average of the probability values. i.e P(Class is Red| X)=0.45. Hence P(Class is Green|X)=0.55. X=Green

# Q3

* 3a.
* library(tree)  
  library(ISLR)  
  attach(Carseats)  
  set.seed(1)  
  train=sample(1:nrow(Carseats),200)  
  Carseats.train=Carseats[train,]  
  Carseats.test=Carseats[-train,]
* 3b.
* tree.carseats=tree(Sales~.,Carseats,subset=train)  
  summary(tree.carseats)
* ##   
  ## Regression tree:  
  ## tree(formula = Sales ~ ., data = Carseats, subset = 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
* par(mfrow=c(1,1))  
    
  tree.pred=predict(tree.carseats,Carseats.test)  
  mean((tree.pred-Carseats.test$Sales)^2)
* ## [1] 4.148897
* plot(tree.carseats)   
   text(tree.carseats)
* 
* Variables that were included in the construction of the tree were: ShelveLoc, Price, Age, Advertising, Income and Comp Price. There are 18 terminal nodes or leaves. The RSS for the training data is 2.36
  + The key variable is ShelveLoc which is split into 2 parts Bad and Medium. ShelfLoc = Good is not a critical indicator of sale price. Price is another key varible which is combined when with ShelveLoc=Bad to give to give split at 120.5 and combined with ShelvLoc=Medium to give a split at price=113. Best Sales number seems to be when the ShelveLoc is a Medium and the sale price is 113.
* the test MSE is 4.15
* 3c.
* cv.carseats=cv.tree(tree.carseats)  
   plot(cv.carseats$size,cv.carseats$dev, type='b')
* 
* prune.carseats=prune.tree(tree.carseats,best=9)  
   plot(prune.carseats)  
   text(prune.carseats, pretty=0)
* 
* yhat=predict(prune.carseats, newdata=Carseats.test)  
   plot(yhat, Carseats.test$Sales, xlab="Predicted Sales Values", ylab="Actual Sales Values")  
   abline(0,1)
* 
* mean((yhat-Carseats.test$Sales)^2)
* ## [1] 4.993124
* Pruning the tree does not reduce the test MSE. Infact the test MSE increases to 4.99
* 3d.
* library(randomForest)
* ## randomForest 4.6-12  
  ## Type rfNews() to see new features/changes/bug fixes.
* set.seed(1)  
  bag.carseats=randomForest(Sales~.,data=Carseats,subset=train,mtry=10,importance=T)  
    
  yhat.bag=predict(bag.carseats,newdata=Carseats.test)  
  plot(yhat.bag, Carseats.test$Sales)  
  abline(0,1)
* 
* mean((yhat.bag-Carseats.test$Sales)^2)
* ## [1] 2.554292
* varImpPlot(bag.carseats)
* 
  + Test MSE after doing bagging is 2.55
  + Price and ShelveLoc are the two most important variables for determining amount of Sales
* 3e.
* set.seed(1)  
  rf.carseats=randomForest(Sales~.,data=Carseats,subset=train,importance=T)  
  yhat.rf=predict(rf.carseats,newdata=Carseats.test)  
  plot(yhat.rf, Carseats.test$Sales)  
  abline(0,1)
* 
* mean((yhat.rf-Carseats.test$Sales)^2)
* ## [1] 3.30763
* varImpPlot(rf.carseats)
* 
  + Test MSE after doing bagging is 3.3 which is higher than bagging but lower than regression tree
  + Price and ShelveLoc are the two most important variables for determining amount of Sales
  + We used the default 'm' for random forest that is i.e = 3 variables. Just using 3 variables increased the Test MSE compared to Bagging which used 10 variables.

# Q4

* 4a.
* library(ISLR)  
  fix(Hitters)  
  Hitters=na.omit(Hitters)  
  Hitter=na.omit(Hitters)  
  Hitter$Salary=log(Hitters$Salary)
* 4b.
* train=1:200  
  test=-train  
  Hitter.train=Hitter[train,]  
  Hitter.test=Hitter[-train,]
* 4c.
* library(gbm)
* ## Loading required package: survival  
  ## Loading required package: splines  
  ## Loading required package: lattice  
  ## Loading required package: parallel  
  ## Loaded gbm 2.1.1
* set.seed(1)  
    
  train.mse=matrix(data=NA, nrow=100, ncol=2)  
  test.mse=matrix(data=NA, nrow=100, ncol=2)  
  for(i in 1:100){  
   lambda=i\*0.01  
   train.mse[i,1]=lambda  
   test.mse[i,1]=lambda  
    
   boost.hitter=gbm(Salary~.,data=Hitter.train, distribution="gaussian", n.trees=1000,interaction.depth=4,shrinkage = lambda)  
   train.mse[i,2]=mean(boost.hitter$train.error)  
    
  yhat.boost=predict(boost.hitter,newdata=Hitter.test,n.trees=1000)  
  test.mse[i,2]=mean((yhat.boost-Hitter.test$Salary)^2)  
  }  
    
  plot(train.mse[,1],train.mse[,2], xlab="Shrinkage factor", ylab="Training MSE")
* 
* 4d.
* plot(test.mse[,1],test.mse[,2], xlab="Shrinkage factor", ylab="Test MSE")
* 
* 4e.
* lm.fit=lm(Salary~.,data=Hitter.train)  
  lm.pred=predict(lm.fit,Hitter.test)  
  mean((lm.pred-Hitter.test$Salary)^2)
* ## [1] 0.4917959
* library(glmnet)
* ## Loading required package: Matrix  
  ## Loading required package: foreach  
  ## Loaded glmnet 2.0-2
* set.seed(1)  
  x=model.matrix(Salary~.,Hitter)  
  y=Hitter$Salary  
  y.test=y[test]  
  grid=10^seq(10,-2,length=100)  
  ridge.mod=glmnet(x[train,],y[train],alpha=0, lambda=grid, thresh=1e-12)  
  ridge.pred=predict(ridge.mod,s=212,newx=x[test,])  
  mean((ridge.pred-y.test)^2)
* ## [1] 0.6313728
  + Minimum Test MSE: Boosting: 0.249
  + Minimum Test MSE: Linear Model fit: 0.49
  + Minimum Test MSE: Ridge Regression: 0.45
* 4f.
* boost.hitter=gbm(Salary~.,data=Hitter.train, distribution="gaussian", n.trees=1000,interaction.depth=4,shrinkage =0.12)  
   summary(boost.hitter)
* 
* ## var rel.inf  
  ## CAtBat CAtBat 21.2500123  
  ## CRuns CRuns 14.1765669  
  ## CWalks CWalks 8.7472089  
  ## CRBI CRBI 7.1509000  
  ## PutOuts PutOuts 6.7109573  
  ## Walks Walks 6.3698013  
  ## AtBat AtBat 5.2080574  
  ## CHmRun CHmRun 4.6710617  
  ## Years Years 4.5471386  
  ## RBI RBI 4.2839575  
  ## Assists Assists 3.9848415  
  ## Hits Hits 2.9274869  
  ## Runs Runs 2.6974009  
  ## HmRun HmRun 2.4462063  
  ## Errors Errors 1.9190240  
  ## CHits CHits 1.0183290  
  ## NewLeague NewLeague 0.8562599  
  ## Division Division 0.7858884  
  ## League League 0.2489012
  + CAtBat and CRuns are the key variables.
* 4g.
* library(randomForest)  
  set.seed(1)  
  bag.Hitter=randomForest(Salary~.,data=Hitter.train,mtry=19,importance=T)  
  yhat.bag=predict(bag.Hitter,newdata=Hitter.test)  
  plot(yhat.bag, Hitter.test$Salary)  
  abline(0,1)
* 
* mean((yhat.bag-Hitter.test$Salary)^2)
* ## [1] 0.228722
  + Test MSE using Bagging: 0.23

# Q5

# Q6

# Q7