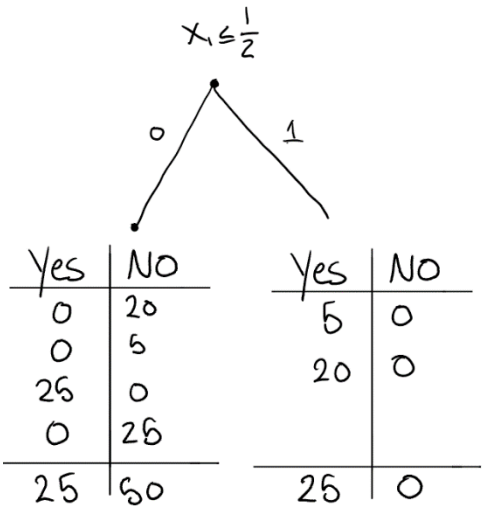
Andrew Glassenbury & Mike Whitley 69657919 & 55503405

Question One

a)



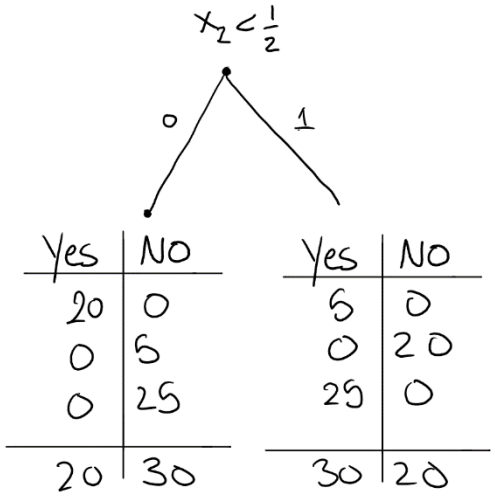
I(m) = ½

I(2m +1) = 2(2/3) = 4/9

I(2m+1) = 0

V(I) = ½ - (¾ *4/9) – (1/4 * 0)

= 18/36 – 12/36 = 6/36 = 1/6



I(m) = ½

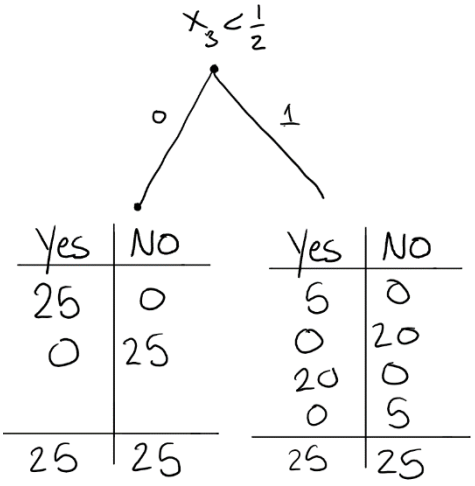
I(2m) = 2(3/5)(1-2/5) = 12/25

I(2m +1) 2(2/5)(1-2/5)

V(I) = ½ - (1/2 * 12/25) – (1/2 * 12/25)

= (1/2) – (12/25) = 1/50

=-



I(m) = ½

I(2m) = 2(1/2) (1-1/2) = 1/2

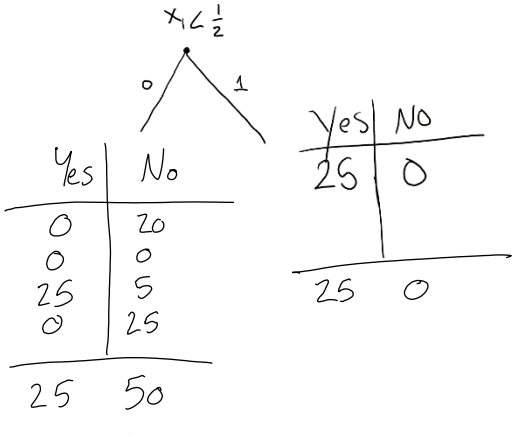
I(2m +1) 2(1/2)(1-1/2) = 1/2

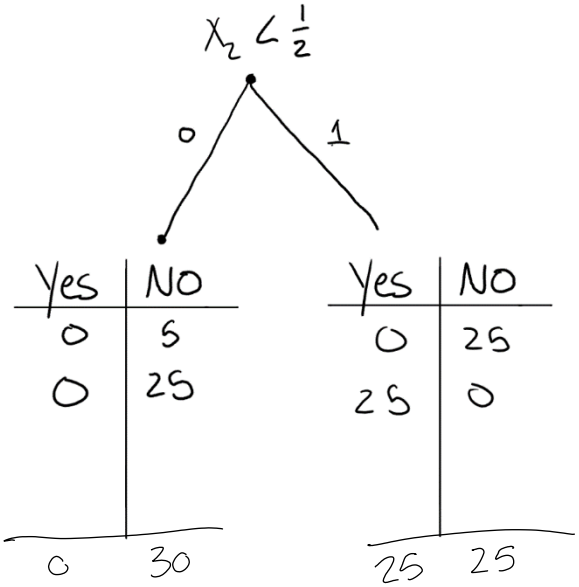
V(I) = ½ - (1/2 * 1/2) – (1/2 * 1/2)

= ½ - ¼ - ¼ = 0

As we want to maximise V(I) so X1 < ½ is best

b)





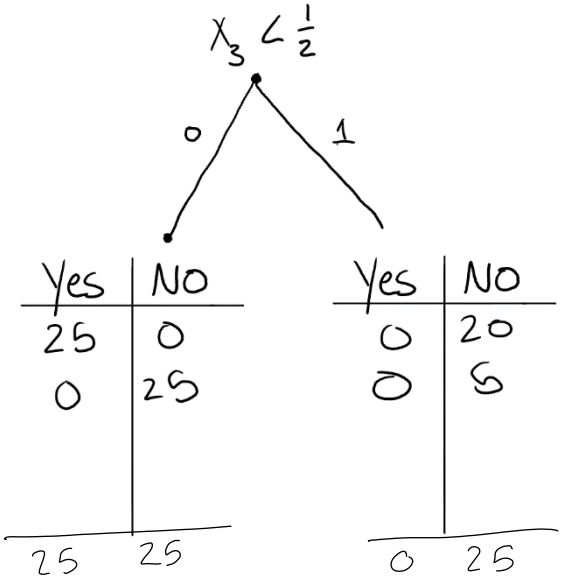
I(m) = ½

I(2m) = 0

I(2m +1) 2(20/45)(1-20/45) = 40/81

V(I) = ½ - (0 * 30/70) – (40/81 * 45/75)

= 11/54



I(m) = ½

I(2m) = 2(25/50)(1-25/50)=1/2

I(2m +1) = 0

V(I) = ½ - (0 * 25/75) – (1/2 * 50/75)

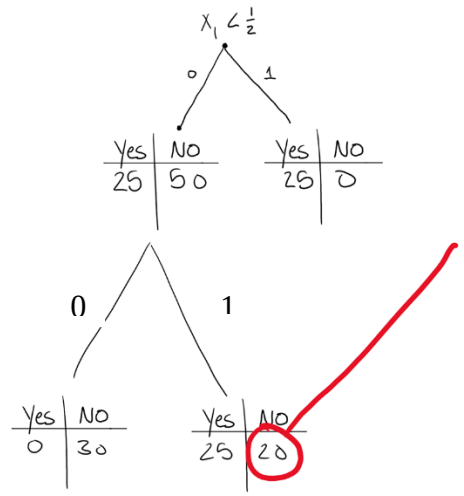
= ½ - 1/3

= 1/6 * 9/54

= 1/36

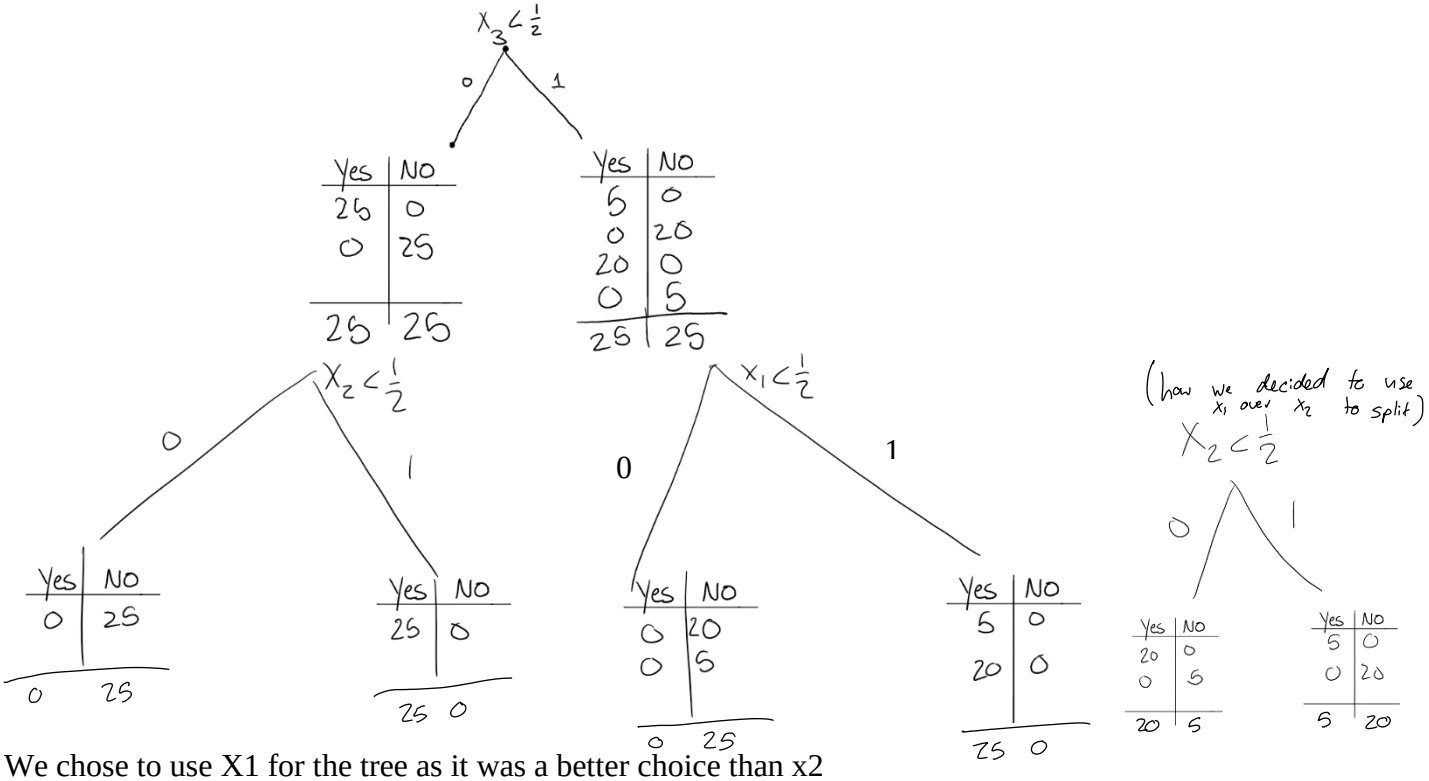
As we want to maximise V(I) we split at x2 < ½

C)



20 are misclassified as No rather than Yes

D)



We chose to use X1 for the tree as it was a better choice than x2

e)

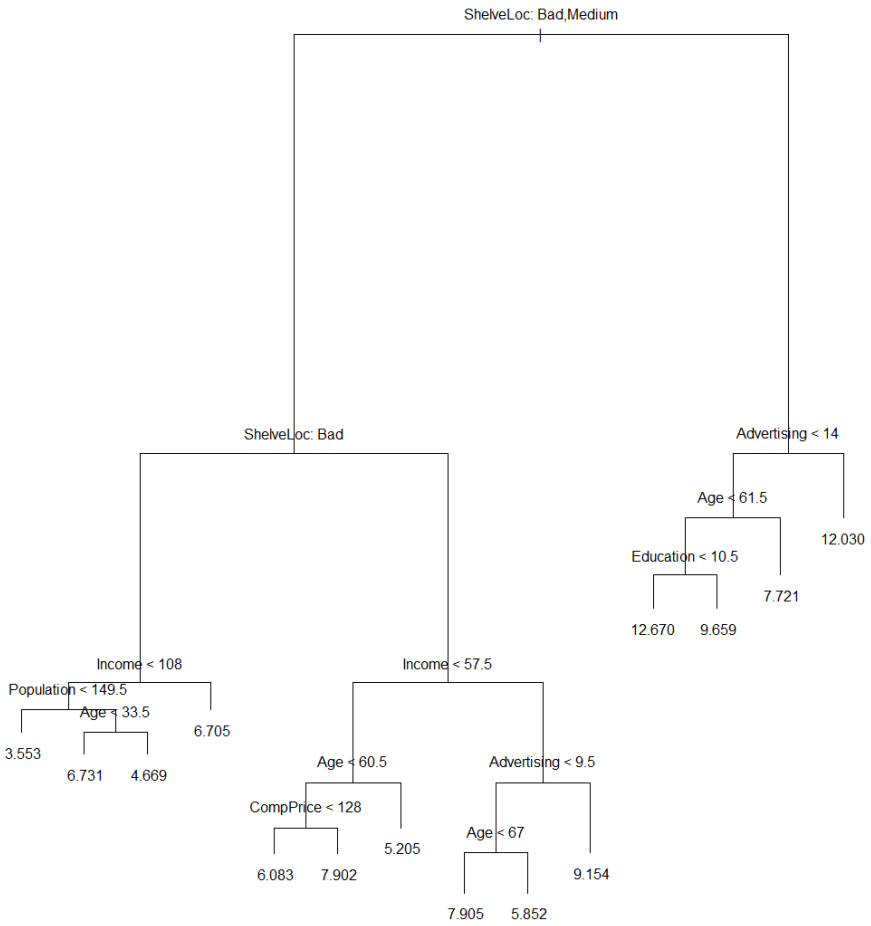
Choosing the local optimal best choice at the time in turn can produce sub optimal trees

With c we end up with misclassified items by splitting locally and in turn produce a sub optimal tree where as by not splitting for local optimal in D we get the global optimal solution

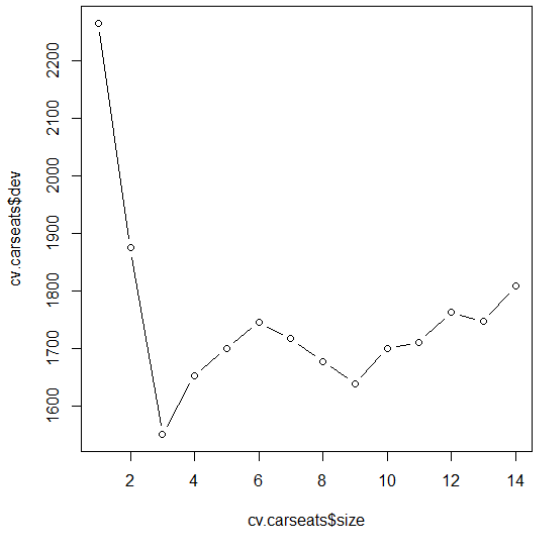
Andrew Glassenbury & Mike Whitley 69657919 & 55503405

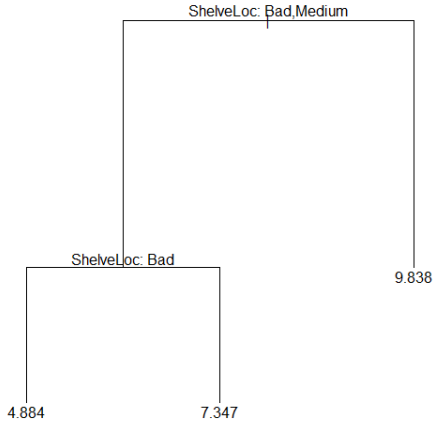
Question Two

a) The tree below shows that the ShelveLoc: Bad,Medium has the greatest reduction of the MSE for the model as it is the largest part of the tree. The terminal nodes show the predicted sales amount for each of the branches.
The training MSE for the tree is 3.521092 and the Test MSE is 5.163211.



b) Using the cv.tree, the best prune to make is at 3. The training MSE is 5.668513 and the Test MSE is 5.346983. The pruning did not improve the Test MSE so the previous tree was better, however the pruned one is less complex and would be easier to use if it was given to someone that was not as capable with statistics.





c) For the bagged tree, it had a Training MSE of 1.108903 and a Test MSE of 4.730355. For the random forest, it had a Training MSE of 1.347235 and a Test MSE of 4.810327. Decorrelating the trees with a random forest has not improved the Test MSE so would not be an effective strategy for this and it would be better to stay with the bagged tree at this point.

d) We found that our best boosted tree had 5000 trees, a depth of 1 and shrinkage =0.001 Training MSE of 4.263347 and a Test MSE of 4.516912. We found this by performing a for loop in R that tested the number of trees as 500, 1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, and 10000. Depth we tested between 1 and 20. Shrinkage we set as 0.01, 0.001, and 0.0001.

e) The model that performed the best was the boosted tree which had the lowest test MSE of 4.516912. The predictors that performed the best in this model was the ShelveLoc variable which had the greatest relative influence on our model predictions of 53.63%.

```
                  var     rel.inf
ShelveLoc     ShelveLoc 53.62629837
Age               Age 18.08770960
Advertising Advertising 15.73533434
Income         Income  8.34721481
Population  Population  2.18202493
CompPrice    CompPrice  1.78456021
Education    Education  0.18058078
US                US  0.05627696
Urban            Urban  0.00000000
```

Andrew Glassenbury & Mike Whitley 69657919 & 55503405

Question Three

a)

X = {d,e}

Y = {a}

Z = {a,d,e}

Total baskets

b) Confidence {d,e} → {a} = support(Z)/ (total baskets contain {d,e}) = 4/6
c) Lift {d,e} → {a} = confidence(Z)/support(XYZ) (4/6)/(4/10) = 5/3

We can see that with a lift value greater than one, the occurrences of {d, e} promotes the increased occurrence of {a} and a reasonable strength of association between {a} and {d,e} i.e. when we see {d,e} its likely to be with {a}. This is supported with the confidence value being showing that 66% of baskets are likely to contain {d,e,a}

a)

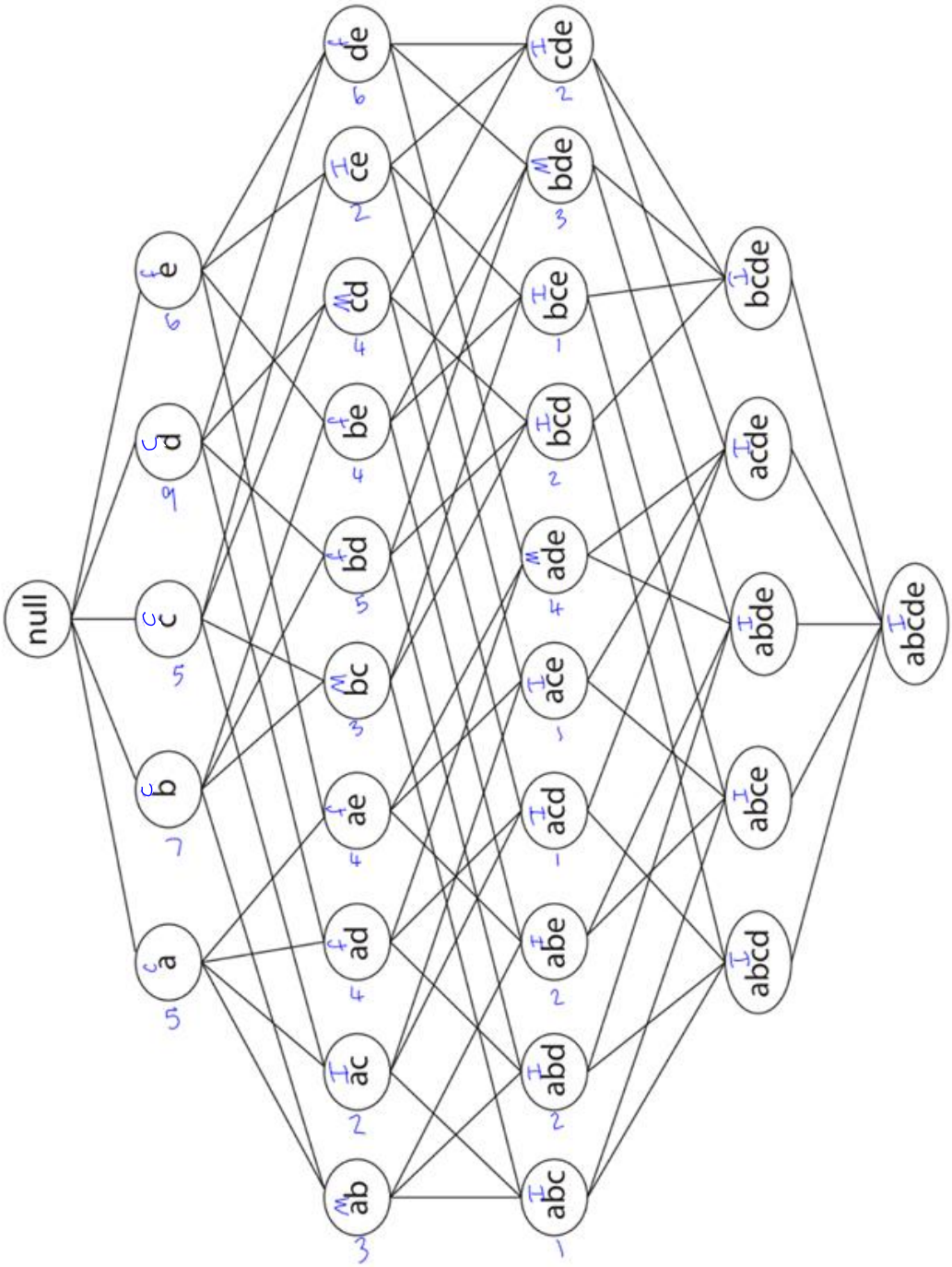


Figure 1: Itemset lattice for Question 3.

R-Code

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

traindata = read.csv('carseatsTrain.csv')
testdata = read.csv('carseatsTest.csv')

##################QUESTION 2#################

################## A

tree.carseats = tree(Sales~., data=traindata)

summary(tree.carseats)

plot(tree.carseats)
text(tree.carseats,pretty=0)

yhat=predict(tree.carseats,newdata=traindata)
Sales.train=traindata[1:266,'Sales']
mean((yhat-Sales.train)^2)


yhat2=predict(tree.carseats,newdata=testdata)
Sales.test=testdata[1:134,'Sales']
mean((yhat2-Sales.test)^2)

#Training MSE is 3.521092
#Test MSE is 5.163211


################## B
cv.carseats = cv.tree(tree.carseats)

plot(cv.carseats$size,cv.carseats$dev,type='b')


prune.carseats = prune.tree(tree.carseats ,best=3)
plot(prune.carseats)
text(prune.carseats ,pretty =0)



yhat3=predict(prune.carseats,newdata=traindata)

mean((yhat3-Sales.train)^2)


yhat4=predict(prune.carseats,newdata=testdata)

mean((yhat4-Sales.test)^2)


#Train MSE 5.668513
#Test MSE 5.346983

#Looking at the plot, the best place to prune the tree is at 3. Doing this, the MSE fo
r both Training and Test MSE have increased making
#it worse.

################## C
tree.carseats = tree(Sales~., data=traindata)
# BAGGED
bag.tree.carseats=randomForest(Sales~.,data=traindata,mtry=3,importance=TRUE)
bag.tree.carseats

yhat.bag.train = predict(bag.tree.carseats,newdata=traindata)

#plot(yhat.bag.train, Sales.train)
#abline(0,1)
mean((yhat.bag.train-Sales.train)^2)
```

```
yhat.bag.test = predict(bag.tree.carseats,newdata=testdata)

mean((yhat.bag.test-Sales.test)^2)

#Train MSE is 0.936179
#Test MSE is 4.889193


# RANDOM FOREST
set.seed(1)
rf.tree.carseats=randomForest(Sales~.,data=traindata,mtry=sqrt(3),importance=TRUE)

yhat.rf.train = predict(rf.tree.carseats,newdata=traindata)

mean((yhat.rf.train-Sales.train)^2)

yhat.rf.test = predict(rf.tree.carseats,newdata=testdata)

mean((yhat.rf.test-Sales.test)^2)

#Train MSE is 1.36193
#Test MSE is 4.766625

#Decorrrellating the trees was not effective in this case because
# both the Train and Test MSE have gotten worse under the
# Random Forest compared to the bagged tree.



############## Boosting ####################################

b=list(500,1000,2000,3000,4000,5000,6000,7000,8000,9000,10000)
b

lambda=list(0.01,0.001,0.0001)
lambda

for (value in b){
  for(value2 in lambda){
    boost.carseats=gbm(Sales~.,data=traindata,distribution="gaussian",n.trees=value,in
teraction.depth=1, shrinkage=value2)
    yhat.boost=predict(boost.carseats,newdata=testdata,n.trees=value)
    print(value)
    print(value2)
    print(paste('TRAIN', (mean((yhat.boost-Sales.train)^2))))
    print(paste('TEST', (mean((yhat.boost-Sales.test)^2))))
    }
}


#b=5000 and lambda=0.001 is the best test outcome.
d=(1:20)

for (value3 in d){
  boost.carseats=gbm(Sales~.,data=traindata,distribution="gaussian",n.trees=5000,inter
action.depth=value3, shrinkage=0.001)
  yhat.boost=predict(boost.carseats,newdata=testdata,n.trees=5000)
  print(value3)
  print(paste('TRAIN', (mean((yhat.boost-Sales.train)^2))))
  print(paste('TEST', (mean((yhat.boost-Sales.test)^2))))
}



#BEST BOOST
boost.carseats=gbm(Sales~.,data=traindata,distribution="gaussian",n.trees=5000,interac
tion.depth=1, shrinkage=0.001)
summary(boost.carseats)

yhat.boost.train=predict(boost.carseats,newdata=traindata,n.trees=5000)
mean((yhat.boost.train-Sales.train)^2)


yhat.boost.test=predict(boost.carseats,newdata=testdata,n.trees=5000)
mean((yhat.boost.test-Sales.test)^2)
```

```
#Train MSE is 4.26308
#Test MSE is 4.506257

#We found the best boosted tree was n.trees=5000,depth=1,
#shrinkage=0.001 . This reduced the test MSE to 4.506257
#


#Boosting performed the best with the lowest MSE for the Test Data.


summary(boost.carseats)
#Best is highest rel.inf
```